ISSN 0542-9951

9 1994

informatyka

P. 1877/94

Wkładka
Systemy czasu rzeczywistego
strony 23–27

WYDAWNICTWO SIGMA NOT

Cena 45 000 zł

informatyka

Miesięcznik
ISSN 0542-9951
INDEKS 36124

nr 9 1994
Wrzesień
Rok wydania XXX

P 1977/84

KOLEGIUM REDAKCYJNE:

mgr Jarosław DEMINET
mgr inż. Piotr FUGLEWICZ
prof. dr hab. Jan GOLIŃSKI
mgr Teresa JABŁOŃSKA
(sekretarz redakcji)
mgr Krystyna
KARWICKA-RYCHLEWICZ
Władysław KLEPACZ
(redaktor naczelny)
mgr inż. Jan RYŻKO
dr Witold STANISZKIS
dr hab. Kazimierz SUBIETA
dr Zdzisław SZYJEWSKI

PRZEWODNICZĄCY
RADY PROGRAMOWEJ:

Prof. dr hab.
Juliusz Lech KULIKOWSKI

WYDAWCA:
Wydawnictwo Czasopism i Książek
Technicznych SIGMA NOT
Spółka z o.o.
ul. Ratuszowa 11
00-950 WARSZAWA
skrytka pocztowa 1004

Redakcja:
01-552 Warszawa,
Pl. Inwalidów 10, p. 104, 105
tel., fax: 39-14-34

Materiałów nie zamówionych
redakcja nie zwraca

W sprawach ogłoszeń prosimy zwracać się bezpośrednio do Redakcji lub Działu Reklamy i Marketingu 00-950 Warszawa ul. Mazowiecka 12 telefon: 27-43-66 telefaks: 26-80-16

W numerze:

	Strona
Edukacja użytkownika – wywiad z dr. Andrzejem Florczykiem – *Krystyna Karwicka-Rychlewicz*	1
I Kongres Informatyki Polskiej	2
Architektura otwartego systemu do syntezy obrazów realistycznych – *Andrzej Wojdała*	3
ARIS – Architektura Zintegrowanych Systemów Informacyjnych – *A.-W. Scheer, T. Elsner, D. Dziuba, M. Lasek*	8
O standaryzacji w zakresie technik kryptograficznych – *Janusz Stokłosa*	13
Podprogramy w języku systemów dynamicznych AMIL – *Marek Stabrowski*	16
Modele zbiorów rozmytych w analizie decyzyjnej przedsiębiorstwa. Część 2 – *Tadeusz Witkowski*	20
SYSTEMY CZASU RZECZYWISTEGO	
Systemy operacyjne czasu rzeczywistego – *Adam Mrózek, Ryszard Winiarczyk*	23

W najbliższych numerach:

W październikowym numerze znajdą się materiały przygotowane na konferencję „Miejskie sieci komputerowe w nauce i gospodarce POLMAN '94".

- **Z prof. Janem Węglarzem rozmawia Krystyna Karwicka-Rychlewicz.**
- **Tomasz Hofmokl, Jan Węglarz i Andrzej P. Wierzbicki omawiają miejskie sieci komputerowe w infrastrukturze informatycznej nauki w Polsce.**
- **Janusz Filipiak, Artur Lasoń i Andrzej R. Pach charakteryzują szerokopasmowe sieci miejskie zgodne ze standardem IEEE 802.6 DQDB.**
- **Daniel Bem i Jarosław M. Janukiewicz zastanawiają się, czy nadszedł już czas na ATM.**
- **Zbigniew Szymański kreśli perspektywy rozwoju transmisji światłowodowej.**
- **Maciej Stroiński naświetla technologie miejskich sieci komputerowych z perspektywy użytkownika i operatora sieci.**
- **Wojciech Burakowski opisuje obsługę ruchu LAN-LAN w sieci B-ISDN.**

Warunki prenumeraty na 1994 r.

Zamówienia na prenumeratę czasopism wydawanych przez Wydawnictwo SIGMA-NOT można składać w dowolnym terminie. Mogą one obejmować dowolny okres czasu, tzn. dotyczyć dowolnej liczby kolejnych zeszytów każdego czasopisma.

Zamawiający może otrzymywać zaprenumerowany przez siebie tytuł począwszy od następnego miesiąca po dokonaniu wpłaty. Zamówienia na zeszyty sprzed daty otrzymania wpłaty będą realizowane w miarę możliwości – z posiadanych zapasów magazynowych.

Warunkiem przyjęcia i realizacji zamówienia jest otrzymanie z banku potwierdzenia dokonania wpłaty przez prenumeratora. Dokument wpłaty jest równoznaczny ze złożeniem zamówienia.

Wpłat na prenumeratę można dokonywać na ogólnie dostępnych blankietach w Urzędach Pocztowych (przekazy pieniężne) lub Bankach (polecenie przelewu), przekazując środki pod adres:
Wydawnictwo SIGMA-NOT Spółka z o.o.
Zakład Kolportażu
00-716 Warszawa, skr. poczt. 1004
konto:
PBK S.A. III 0/Warszawa nr 370015-1573-139-11

Wpłaty na prenumeratę przyjmują także wszystkie urzędy pocztowe nadawczo-odbiorcze oraz doręczyciele na terenie całego kraju

Na blankiecie wpłaty należy czytelnie podać nazwę zamawianego czasopisma, liczbę zamawianych egzemplarzy, okres prenumeraty oraz własny adres.
Na życzenie prenumeratora, zgłoszone np. telefonicznie, Zakład Kolportażu ul. Bartycka 20, 00-950 Warszawa, (telefony: 40-30-86, 40-35-89 oraz 40-00-21 wew. 249, 293, 299) wysyła specjalne blankiety zamówień wraz z aktualną listą tytułów i cennikiem czasopism.

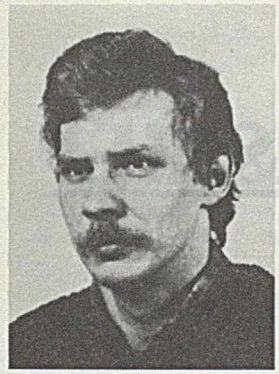
Odbiorcy zagraniczni mogą otrzymywać czasopisma poprzez prenumeratę dewizową (wpłata dokonywana poza granicami Polski w dewizach, wg cennika dewizowego z cenami podanymi w dolarach amerykańskich) lub poprzez zamówioną w kraju prenumeratę ze zleceniem wysyłki za granicę (zamawiający podaje dokładny adres odbiorcy za granicą, dokonując równocześnie wpłaty w wysokości dwukrotnie wyższej niż cena normalnej prenumeraty krajowej).

Egzemplarze archiwalne (sprzedaż przelewowa lub za zaliczeniem pocztowym) można zamawiać pisemnie, kierując zamówienia pod adresem: Wydawnictwo SIGMA NOT, Spółka z o.o. Zakład Kolportażu, 00-716 Warszawa, ul. Bartycka 20, paw. B, tel. 40-37-31, natomiast za gotówkę można je nabyć w Klubie Prasy Technicznej w Warszawie ul. Mazowieckiej 12, tel. 26-80-17.

Istnieje możliwość zaprenumerowania 1 egz. czasopisma po cenie ulgowej przez indywidualnych członków stowarzyszeń naukowo-technicznych zrzeszonych w FSNT oraz przez uczniów zawodowych i studentów szkół wyższych. Blankiet wpłaty na prenumeratę ulgową musi być opatrzony na wszystkich odcinkach pieczęcią koła SNT lub szkoły.

W przypadku zmiany cen w okresie objętym prenumeratą Wydawnictwo zastrzega sobie prawo do wystąpienia o dopłatę różnicy cen oraz prawo do realizowania prenumeraty tylko w pełni opłaconej.
Cena jednego egzemplarza: normalna 45 000 zł, ulgowa 31 500 zł
Wartość prenumeraty w zł:
Normalna: kwartalna 135 000, półroczna 270 000, za trzy kwartały 404 000
Ulgowa: kwartalna 94 500, półroczna 189 000, za trzy kwartały 283 500

Skład i druk: Drukarnia SIGMA NOT Sp. z o.o. z. 316/94



Edukacja użytkownika

Z dr. Andrzejem Florczykiem prezesem Stowarzyszenia Rozwoju Systemów Otwartych rozmawia Krystyna Karwicka-Rychlewicz

– W mijającym roku dość wyraźnie zostały określone cele i zakres działania trzech spośród czterech organizacji grupujących osoby i firmy związane z informatyką. PTI, najstarsza z organizacji, jest stowarzyszeniem naukowym, deklaruje też, że staje się reprezentantem interesów ludzi wykonujących zawody informatyczne. Stowarzyszenie Polski Rynek Oprogramowania PRO – dba o interesy twórców oprogramowania. PIIiT jest samorządową organizacją gospodarczą i reprezentuje dostawców technologii. Czy równie jasno zostały już sprecyzowane cele i sposób działania Stowarzyszenia Rozwoju Systemów Otwartych?

– Nasze Stowarzyszenie, również na mocy porozumień z trzema pozostałymi organizacjami stara się reprezentować interesy coraz szerszej rzeszy użytkowników informatyki. Do tej pory nikt poważnie nie myślał o tym, że interesy te są często sprzeczne z interesami dostawców technologii, a nieraz i z interesami środowiska zawodowego informatyków. Nie do końca jeszcze wiemy, jak to robić, ale po wstępnych dyskusjach, po spotkaniach w Mrągowie, po stworzeniu założeń I Kongresu Informatyki Polskiej, mamy już pełniejszy obraz celów i sposobów działania SRSO.

– A czy użytkownik informatyki w Polsce jest już na tyle świadomy, że potrafi sprecyzować swoje interesy i dostrzega potrzebę ich obrony?

– To istota sprawy. Myślę, że dzisiaj nie są jeszcze dobrze wykrystalizowane interesy grupowe użytkowników informatyki. O ile federacje konsumentów np. elektroniki, już określiły podstawowe cele, pilnują by telewizor czy magnetofon spełniały normy bezpieczeństwa i inne parametry, o tyle w przypadku dóbr informatycznych dopiero rodzą się takie wymagania. Dlatego też SRSO spełnia dzisiaj głównie funkcje edukacyjne. Chcemy, by użytkownik nie tylko mógł kupować sprzęt dobry, ale i dysponował systemami, które wymieniają łatwo ze sobą informacje. Stąd zresztą pochodzi nazwa Stowarzyszenia. Chcemy promować ułatwienia w zakresie wymiany informacji, wymuszać je na producentach. Przypuszczam, iż wkrótce zaczniemy w Polsce przeżywać wielkie, gorące, spory wokół ustawy o ochronie dóbr osobistych. Jak wiemy bowiem z doświadczeń innych, chociażby amerykańskich, informatyka niesie w tej dziedzinie wiele niebezpieczeństw. Otwartość systemów informatycznych może być w pewnym momencie swoiście rozumiana, na przykład administracja zechce zmonopolizować dużą część informacji personalnej. Pojedyncze informacje nie stanowią zagrożenia, ale zmasowane w rękach urzędników mogą stać się straszliwą bronią. Wizja, nazwijmy ją Orwellowską, to jeden z następnych elementów, który stwarza perspektywę rozwoju Stowarzyszenia. Jego rola polegałaby bowiem nie tylko na ustalaniu grupowych kryteriów w stosunku do technologii, ale i stosunku do społecznych skutków informatyzacji społeczeństwa. Oczywiście, obecnie jesteśmy na razie na etapie dbania o interesy użytkowników w inny sposób: staramy się upowszechniać świadomość, że informatyka jest tylko środkiem zaspokajania pewnych potrzeb, ułatwienia pracy, zwiększenia jej efektywności. I że narzędzie informatyczne musi być dostrzegane w szerszym kontekście. Ale rozwiązania, które zaczynają już istnieć jako fragment rozwoju światowej informatyki – w Polsce uległy pewnym opóźnieniom fazowym. Dlatego dopiero teraz zaczyna istnieć grupa świadomych użytkowników, którzy chcą mieć perspektywiczny program rozwoju informatyki. Wydaje mi się, że nie przypadkiem historia Biura do Spraw Informatyki URM i Stowarzyszenia są tak skorelowane, powiązane ze sobą, również personalnie. Ja reprezentuję bowiem jednocześnie interesy ogromnej i bardzo ważnej grupy użytkowników.

– Rzeczywiście, Pan jako wieloletni dyrektor Biura do Spraw Informatyki URM, reprezentuje ogromną grupę użytkowników...

– ...I apeluję do nich, czyli do urzędników administracji państwowej, by zapisywali się do Stowarzyszenia. W urzędach nie możemy sobie pozwolić na wychodzenie poza przepisy, Stowarzyszenie natomiast jest forum dyskusyjnym, mam nadzieję, dającym początek silnego ruchu lobbystycznego. Chcemy pomóc użytkownikom w zdobywaniu wiedzy, która ułatwi lepsze definiowanie wymagań i potrzeb w stosunku do systemu.

– Czy sprawi także, że potrafi on lepiej bronić się przed nieuczciwymi dostawcami sprzętu, oprogramowania i twórcami systemów? Inaczej mówiąc, czy Stowarzyszenie chce być również organizacją konsumencką?

– W pewnym sensie tak, ale musi to być organizacja konsumencka innego typu niż w pozostałych branżach. Tu nie chodzi tylko o to, by sprzedawano nam dobre technologie. Przeżywamy obecnie śmierć kolejnego mitu. Na początku informatyzację miały załatwić wielkie, scentralizowane systemy; nic nie załatwiły, bo życie generowało nowe potrzeby. Później PC miał być uniwersalnym narzędziem i lekarstwem na wszystkie bolączki. Następny etap – rozbudowa sieci lokalnych, umożliwiających lepszą organizację pracy małych grup. Dzisiaj jesteśmy na nowym etapie: PC odzyskał swój normalny wymiar, przestał być dostrzegany jako panaceum na wszystko, jest tylko wejściem w świat rozległych sieci. Mają być one radykalną metodą porozumiewania się, wymiany i przekazywania przetworzonej informacji. To prawda, ale chodzi o świadome używanie tych systemów. I przygotowanie użytkowników do tego, by umieli korzystać z informacji w sposób racjonalny. Naczelnym zagadnieniem jest stworzenie takiego modelu edukacji użytkowników informatyki, by wiedzieli do czego system ma służyć. Jeszcze parę lat temu burmistrz, gdy miał w gabinecie terminal lub PC mówił, że się skomputeryzował. Obojętne było, co on z nim robił. Dzisiaj już sam fakt, że decyzja o informatyzacji danego urzędu jest uwikłana w proces związany z pracą nad modelem informacyjnym państwa, nad przestrzeganiem pewnych standardów jakościowych, technologicznych i wymiany danych powoduje, że SRSO znalazło swoje miejsce. Nie przypadkiem na zjazdy Stowarzyszenia do Krakowa, na forum do Łańska (gdzie w tym roku, na żądanie polityków, odbywa się drugie, jesienne, forum), na których najwyżsi przedstawiciele rządu prezentują strategię rozwoju informatyki w państwie, przyjeżdża po kilkaset osób. Opracowywane są decyzje rządowe i parlamentarne, ruszą zespoły, które będą przygotowywały cały pakiet ustaw, istotnych dla zapewnienia interesów użytkownika, a nie firm czy twórców. Trwają prace nad ustawą o ochronie dóbr osobistych w systemach informatycznych, nad przepisami dotyczącymi informatycznych rejestrów państwowych, przyspieszana jest debata na temat ustawy o statystyce, której duża część dotyczy informatyki. Muszę też wspomnieć o gorących sporach dotyczących ustawy o tajemnicy państwowej i roli systemów komputerowych. Bez wątpienia stanowisko Stowarzyszenia zyska bardzo wysoką rangę w debacie społeczno-politycznej na temat tych ustaw, a szczególnie ustawy o ochronie dóbr osobistych.

– Czy w związku z określeniem celów i roli Stowarzyszenia wzrasta również zainteresowanie przynależnością do niego?

– Od majowego zjazdu w Łańsku nie nadążamy z rejestracją nowych członków. Ze 120 ich liczba wzrosła do kilkuset. Mam na myśli tych, którzy regularnie opłacają składki. Do tego dołączają sponsorujące firmy komputerowe i telekomunikacyjne.

– Firmy sponsorujące? Jak można pogodzić dbałość o interesy użytkowników, z braniem pieniędzy na działalność Stowarzyszenia od dostawców technologii informatycznych!

– Zakłada pani niecne zamiary firm. Tymczasem gra rynkowa, a zwłaszcza szalona konkurencja wymusza dobre intencje obu stron. To bardzo pozytywny objaw. Firmy, które są blisko użytkownika, a Stowarzyszenie stwarza platformę bliskich kontaktów, mogą łatwiej do-

dokończenie na s. 12

I KONGRES INFORMATYKI POLSKIEJ

POZNAŃ
1–3 grudnia 1994 r.

Komitet Honorowy pod przewodnictwem Pana Premiera Waldemara Pawlaka

Komitet Programowo-Organizacyjny

Przewodniczący
Wacław Iszkowski

Wiceprzewodniczący
Andrzej Florczyk – Przewodniczący Komisji Użytkowników
Piotr Fuglewicz – Przewodniczący Komisji Edukacji i Szkoleń
Marek Maniecki – Przewodniczący Komisji Producentów i Dostawców

Szefowie Zespołów
Programowego – Dariusz Kupiecki
Finansowego – Marcin Bańkowski
Promocyjnego – Jarosław Deminet

Członkowie
Witold Abramowicz, Tadeusz Beliwar, Sławomir Chabros, Roman Dolczewski, Bogusław Dziarnowski, Franciszek Florek, Andrzej Gogolewski, Adam Kowalski, Robert Liwiński, Bogdan Michalak, Tomasz Niesłuchowski, Jerzy Nowak, Jacek Stochlak, Zdzisław Szyjewski

Do środowiska informatycznego w Polsce

Wkroczyliśmy w erę informatyzacji – nadszedł więc czas, aby wspólnie w całym polskim środowisku informatycznym przeanalizować stan obecny informatyki polskiej oraz jej możliwy rozwój, szczególnie ten, który może być dokonany naszymi siłami.

Kongres ma być podsumowaniem wielu analiz i dyskusji odbywających się w ostatnim roku oraz zgromadzić materiały robocze do przygotowania Raportu o strategii rozwoju informatyki w Polsce w najbliższych latach. Raport ten, powstający w całym środowisku informatycznym, ma prezentować szanse, konieczne zmiany, ma podkreślać nasze wspólne interesy, ale i też nie kryć naturalnych sprzeczności. Raport przedstawiony władzom ma być naszymi oczekiwaniami na racjonalne sterowanie rozwojem informatyki w Rzeczypospolitej Polskiej.

Zapraszamy do udziału
Każdy głos się liczy

Komitet Programowo-Organizacyjny
I Kongresu Informatyki Polskiej

Program Kongresu

1 grudnia 1994 – dzień „0":

16:00 Początek rejestracji uczestników
19:00 Otwarcie Kongresu
20:00 Cocktail

2 grudnia 1994 – dzień „1":

9:00 Sesja plenarna
10:00 Sesje robocze w grupach tematycznych
A. Rządowe przedsięwzięcia informatyczne
B. Warunki rozwoju polskiego oprogramowania
C. Świadomość społeczna w odbiorze informatyki
D. Wypełnianie potrzeb kadrowych w informatyce
E. Powiązania między rozwojem telekomunikacji i informatyki
F. Nowe technologie informatyczne
15:00 Sesje robocze w grupach środowiskowych
UZ – Użytkownicy
ES – Edukacja i Szkolenia
PD – Producenci i Dostawcy
20:00 Cocktail

3 grudnia 1994 – dzień „2":

9:00 Sesja plenarna z udziałem gości honorowych Kongresu
Podsumowanie wyników z sesji roboczych w grupach
14:00 Zakończenie Kongresu

INFORMACJE ORGANIZACYJNE

Kongres odbędzie się w dniach 1–3 grudnia 1994 r. na terenie Międzynarodowych Targów Poznańskich (paw. 28).

Warunkiem uczestnictwa jest przysłanie DEKLARACJI UCZESTNICTWA (wydrukowanej na s. 14), w terminie do 31 października 1994 r.

Podstawą udziału w Kongresie jest otrzymanie od organizatorów KARTY UCZESTNIKA, która będzie wysyłana do zgłoszonych osób po:
- kwalifikacji przez Komitet Programowy (istotna będzie kolejność zgłoszeń),
- opłaceniu kosztów uczestnictwa na podstawie otrzymanej od organizatorów faktury.

Koszt uczestnictwa obejmuje: udział w obradach, materiały, świadczenia recepcyjne, obiady w dniach 2–3 grudnia, udział w cocktailach.

Wpłata kosztów uczestnictwa powinna nastąpić nie później niż do 20 listopada 1994 r.

Koszt uczestnictwa nie obejmuje: kosztów dojazdu do/z Poznania oraz kosztów zakwaterowania w hotelu.

Organizatorzy dokonali rezerwacji miejsc w następujących hotelach Poznania (uzyskując wysokie upusty w cenach pokoi):

	Koszt jednej doby w pokoju (ze śniadaniem) jednoosobowym	dwuosobowym
Hotel POZNAŃ****	830.000 + 22% VAT	580.000 + 22% VAT
Hotel MERKURY***	570.000 + 7% VAT	400.000 + 7% VAT
Hotel PAN	500.000 + 7% VAT	375.000 + 7% VAT

Rezerwacja miejsc hotelowych na załączonej DEKLARACJI UCZESTNICTWA.

Pełne informacje o programie I KONGRESU INFORMATYKI POLSKIEJ oraz uzupełniające informacje organizacyjne Uczestnicy Kongresu otrzymają w KARTACH UCZESTNIKA.

BIURO ORGANIZACYJNE
00-503 WARSZAWA 15, skr. 44
ul. Żurawia 4a, p. 200
tel. (02) 621-76-26, 693-59-46
komertel 39121125, fax (02) 693-59-49

Deklaracja uczestnictwa na s. 15

Architektura otwartego systemu do syntezy obrazów realistycznych

Andrzej Wojdała
Instytut Informatyki i Automatyki Morskiej
Politechnika Szczecińska

W ciągu ostatnich kilku lat synteza realistycznych obrazów zyskała powszechną akceptację w wielu dziedzinach projektowych, stając się najbardziej dynamicznie rozwijanym obszarem grafiki komputerowej. Z roku na rok pojawiają się zarówno nowe metody i algorytmy, jak i nowe rozwiązania znanych problemów; rosną też możliwości sprzętu graficznego. Zasadniczym trendem jest symulacja zjawisk fizycznych, zachodzących przy propagacji światła w środowisku, oraz przy odbiciu światła od powierzchni obiektów. Synteza realistycznych obrazów stała się dziedziną interdyscyplinarną, czerpiącą z osiągnięć fizyki, optyki, kolorymetrii, teorii sygnałów, statystyki, metod optymalizacji, psychofizjologii widzenia i innych.

Praca badawcza i dydaktyczna w zakresie syntezy obrazów najczęściej wymaga zastosowania nowo opracowywanych algorytmów w połączeniu z już istniejącymi. W przeciwnym wypadku nieunikniona jest praca w zespołach, w których działalność jednostki musi być ściśle koordynowana, lub powielanie wykonanej już raz pracy, często w zakresie zagadnień rutynowych dla syntezy obrazu.

Przedmiotem artykułu jest zaprezentowanie architektury takiego systemu, który raz zbudowany i posiadający stałą opiekę pozwoli na samodzielną pracę badawczą i dydaktyczną w zakresie syntezy obrazu.

Podstawowe cele tego systemu to:

- Skrócenie czasu pracy nad implementacją nowych algorytmów, a więc wyeliminowanie konieczności opracowywania zagadnień, które nie są bezpośrednim przedmiotem badań. Ponadto, należy uwzględnić specyfikę środowisk akademickich, gdzie część prac jest wykonywana przez studentów, a więc występuje stała rotacja wykonawców. Wymaga to struktury hierarchicznej i silnie zmodularyzowanej, pozwalającej stałej obsłudze na łatwe zarządzanie systemem.
- Uwzględnienie przewidywanych w następnych latach kierunków rozwoju syntezy obrazu, w szczególności opracowywanie kompletnych, nowych modeli oświetlenia i odbicia, a także obsługa bardzo dużych baz danych, wielokrotnie większych niż dostępna pamięć.
- Umożliwienie oceny jakości nowych metod przez porównanie ich wyników z wynikami algorytmów już dostępnych w systemie. Należy stwierdzić, że obiektywne porównanie wydajności poszczególnych algorytmów jest w zasadzie możliwe tylko w warunkach takiego systemu. Porównania algorytmów osadzonych w różnych systemach nie są wiarygodne, gdyż są obarczone błędem wynikającym z innego oprogramowania zagadnień nie będących przedmiotem badań.
- Możliwość oceny wydajności algorytmów przez zapewnienie statystyki monitorowania wywołań poszczególnych modułów oraz pomiar czasu.
- Umożliwienie oceny jakości nowych metod przez porównanie ich wyników z danymi pomiarowymi i obliczeniami analitycznymi.
- Wykorzystanie rosnących możliwości sprzętu graficznego w celu wspomagania syntezy obrazu, w tym wykorzystanie przetwarzania równoległego. Architektura systemu powinna jednocześnie umożliwiać pokonanie algorytmicznych „wąskich gardeł" przez konstruowanie dedykowanego sprzętu, dostępnego przez odpowiednie moduły programowe.
- Zapewnienie maksymalnej przenośności całego systemu, pomimo intensywnego wykorzystywania możliwości sprzętu.
- Zapewnienie kompatybilności danych w miarę rozwoju systemu przez przyjmowanie standardowych formatów danych wejściowych, opracowanie rozbudowywalnych formatów pośrednich i generowanie wyników w standardowych formatach wyjściowych.

Przegląd opublikowanych rozwiązań

Od czasu opublikowania przez Whitteda [75] struktury systemu badawczego do syntezy obrazu opracowanego w Bell Laboratories, mało elastycznego zarówno na globalnym, jak i na lokalnym poziomie, w literaturze opisano wiele podobnych systemów, o celach przynajmniej częściowo zbieżnych z przedstawionymi na wstępie.

Znane z literatury rozwiązania można podzielić na trzy kategorie [69]:

- systemy monolityczne, sterowane za pomocą zbiorów kontrolnych lub opcji wykonania,
- zestaw modułów, konfigurowalnych przez definiowanie przepływu pomiędzy nimi danych,
- biblioteki, wykorzystywane do tworzenia aplikacji wyższego poziomu.

Systemy monolityczne w oczywisty sposób nie spełniają większości postawionych wyżej celów. Ich funkcjonalność bywa różna, ale jest trudno rozbudowywalna. System REYES (*Renders Everything You Ever Saw*) opracowany w Lucasfilm [15] jest zorientowany w stronę jednej, konkretnej techniki syntezy obrazu, jest zatem systemem zamkniętym. BAGS (*Brown Animation Generation System*), używany w Uniwersytecie Brown [64] obejmuje wiele metod syntezy obrazu, oraz dzięki wielkości modułów programowych umożliwia opracowywanie nowych technik; niestety, wzajemne wielorakie powiązania między modułami praktycznie ograniczają te możliwości. Monolityczne systemy RADIANCE [53] i RayShade [54] dostępne jako oprogramowanie „public domain", mają podobne wady, które są jednak częściowo skompensowane powszechnością i dostępnością postaci źródłowej tych systemów, co zostanie jeszcze skomentowane w podsumowaniu.

Pierwsze systemy złożone z luźno połączonych, konfigurowalnych modułów zostały opisane przez Crowa [16] i Blinna [8]. Podobnie zaprojektowany FRAMES (*Flexible Rendering, Animation and Modeling Experimentation System* [51]), zrealizowany jako zestaw filtrów dla systemu operacyjnego UNIX, dobrze nadaje się do eksperymentów z obliczeniami rozproszonymi, ale umożliwia zastosowanie tylko prostych technik syntezy obrazu (cieniowanie Gouraud, Phonga). Ewentualne włączenie technik śledzenia promieni i bilansu promieniowania przerzuca cały wysiłek dotyczący

* Tłumaczenie artykułu: „An Open image synthesis system architecture", opublikowanego w: Proceedings of the 5-th School Microcomputer'94, Computer Vision and Graphic, edited by Wojciech Zamojski, Dariusz Caban; Wydawnictwo Prac Naukowych FORMAT, Wrocław 1994

opracowania architektury odpowiedniego filtru na implementatora. Podobny zarzut można wysunąć pod adresem systemu GRAPES (*General Rendering Applications Programming Environment* [42]).

Pod względem możliwości rozbudowy podobnie prezentuje się system IRIS Explorer [62], służący do szeroko pojętej wizualizacji. System składa się z dużej liczby modułów, dowolnie konfigurowalnych w tzw. mapy. Mapy te są w istocie aplikacjami wyższego poziomu, mogącymi służyć np. do syntezy obrazu. Możliwa jest budowa własnych modułów, co zwiększa elastyczność systemu. Wadą jest możliwość budowania aplikacji tylko na wysokim poziomie. Wymiana algorytmów niższego poziomu wymaga zbudowania własnego podsystemu (nowych modułów spełniających rolę systemu syntezy obrazu, zatem IRIS Explorer spełnia postawione cele tylko w zakresie wyeliminowania czynności rutynowych, jak np. sama wizualizacja.

Na pograniczu kategorii systemów o budowie modułowej i bibliotecznej lokuje się system do celów badawczo-rozwojowych, opracowany w Laboratoriach Syntezy Obrazu i Animacji Komputerowej (MIRALab) Uniwersytetu w Montrealu [67]. System ten jest udostępniany użytkownikowi jako strukturalny język programowania do celów syntezy realistycznych obrazów i animacji (MIRA-SHADING), oraz język interakcyjnej syntezy i animacji (INTERMIRA). Aplikacje wyższego poziomu są tworzone jako programy w tych językach (np. system syntezy obrazów realistycznych SABRINA, opracowany tamże), są zatem ograniczone ich zakresem. Innymi słowy, dodawanie nowych algorytmów i metod wymaga rozszerzeń języków, co z punktu widzenia typowego użytkownika nadaje im dodatkowo cechy systemów zamkniętych.

Najliczniejszą grupę stanowią systemy zbudowane jako biblioteki procedur (lub biblioteki obiektów, w przypadku obiektowej koncepcji systemu). Najpowszechniej znanym systemem o strukturze biblioteki procedur jest RenderMan, opracowany w firmie Pixar [69]. System tem dostarcza metodologii opisu sceny: obiektów, źródeł światła i kamery. Istotną koncepcją jest oddzielenie modelowania od samej syntezy obrazu, co jest zbieżne z postawionym celem dotyczącym formatu pośredniego danych. Użytkownik pisze i uruchamia program w konwencjonalnym języku, wywołując zestaw procedur bibliotecznych. W sposób najbardziej elastyczny potraktowano opis wyglądu obiektu, który może być definiowany w specjalnym języku (*Shading Language*). Procedury napisane w tym języku są wywoływane w czasie syntezy obrazu, zatem postulat łatwości implementacji nowych modeli odbicia jest całkowicie spełniony. Z drugiej strony, RenderMan wykazuje sztywność jeśli chodzi o wymianę wewnętrznych algorytmów, takich jak techniki przyspieszania obliczeń (np. podział przestrzeni), co uniemożliwia zastosowanie go jako funkcjonalnie pełnego narzędzia do celów badawczych i dydaktycznych. Podobne wady ma zestaw uzupełniających się pakietów bibliotecznych opracowanych przez Silicon Graphics: IRIS Graphics Library (GL) [61], IRIS Inventor 3D Toolkit [65] oraz IRIS Performer [63]. Z drugiej strony, konstrukcja tych bibliotek umożliwia postulowaną obsługę dużych baz danych i skorzystanie z możliwości oferowanych przez sprzęt, niestety kosztem przenośności (chociaż OpenGL powoli staje się de facto standardem biblioteki graficznej.

Jak można zauważyć, żaden z przedstawionych systemów nie spełnia całkowicie postawionych celów badawczo-dydaktycznych. Najbardziej dojrzałe pod tym względem wydają się: SPECTRUM oraz system opracowany w Uniwersytecie Cornell, które należy zaliczyć do systemów bibliotecznych. SPECTRUM [22] stawia sobie cele podobne do wymienionych na wstępie: nauczanie w zakresie technik syntezy obrazu, szybkie opracowywanie nowych algorytmów bez konieczności troszczenia się o pozostałe aspekty syntezy obrazu. SPECTRUM jest podzielony na siedem podsystemów; niestety, podstawowy podsystem SpectrumBasic stanowi zestawienie procedur i typów dotyczących wielu różnych zagadnień, często dość odległych: dotyczących geometrii (np. bryły otaczające, przecięcie bryły z promieniem), śledzenia promienia, próbkowania i rekonstrukcji, statystyki, modelu odbicia (tu zaczerpnięto z systemu RenderMan pomysł programowanego modelu odbicia). Autor systemu uzasadnia takie przemieszanie wzajemnymi ścisłymi powiązaniami zagadnień, co jest niewątpliwie prawdą, ale rodzi niebezpieczeństwo ograniczenia elastyczności systemu przez konieczność znajomości tych relacji (metodologię projektowania systemu o dobrze odseparowanych algorytmach przedstawili Kirk i Arvo [34], a następnie Arvo [3]; niestety dotyczy ona tylko technik przyspieszających w metodzie śledzenie promieni. Z drugiej strony, niewątpliwą zaletą SPECTRUM jest zapowiedziana dostępność kodu źródłowego.

System stosowany w Uniwersytecie Cornell [68], będący rozwinięciem wcześniejszego systemu badawczego [27], uniknął niebezpieczeństwa zbyt wielu powiązań pomiędzy poszczególnymi algorytmami, przez udaną hierarchizację i modularyzację zagadnień syntezy obrazu. Także cele systemu są bardzo zbieżne z zaprezentowanymi. Wydaje się jednak, że stopień hierarchizacji jest zbyt niski, co powoduje pewne przemieszanie zagadnień, choć nie tak silne jak w SPECTRUM. Do niedostatków tego systemu należy też zaliczyć brak troski o uwzględnienie w opracowywanych algorytmach rosnących możliwości popularnego sprzętu graficznego. W szczególności, poza enigmatycznie zasygnalizowanym modułem równoległego bilansu promieniowania (ang. *parallel radiosity*), przetwarzanie równoległe zostało sprowadzone do wysokiego poziomu (połączony w sieć zestaw stacji graficznych), co utrudnia zrównoleglanie poszczególnych algorytmów poziomów niższych, a także wykorzystanie coraz powszechniejszych maszyn wieloprocesorowych. Ponadto, nie została przewidziana możliwość eksperymentowania z dedykowanym sprzętem graficznym w celu usuwania algorytmicznych „wąskich gardeł".

Założenia dotyczące sprzętu i środowiska programowego

Postulowane na wstępie cele wymagają zastosowania stacji graficznych pracujących w systemie UNIX, co jest umotywowane względami zarówno programowymi, jak i sprzętowymi.

Względy sprzętowe, to nie tylko moc obliczeniowa, ale przede wszystkim możliwość spełnienia postulatu wspomagania syntezy obrazu przez dostępność niektórych operacji graficznych, realizowanych sprzętowo (np. *z*-bufor, cieniowanie metodą Gouraud i Phonga, perspektywa powietrzna). Jednocześnie, stacje graficzne są często dostępne w wariancie wieloprocesorowym z pamięcią wspólną, łatwym do obsługi z poziomu systemu operacyjnego UNIX [76].

System UNIX to możliwość pracy rozproszonej (a więc możliwość zrealizowania przetwarzania równoległego na poziomie węzłów), przy czym węzłami mogą być również komputery osobiste. Jest to istotna cecha, gdyż umożliwia eksperymenty z dedykowanym sprzętem, jako że rozbudowa komputera osobistego jest daleko prostsza niż rozbudowa stacji graficznej. Wielozadaniowość systemu umożliwia łatwą symulację algorytmów równoległych [76]. UNIX upraszcza też zrealizowanie postulatu prowadzenia statystyki i monitorowania wywołań, przez zastosowanie narzędzi dostępnych w tym systemie. Również postulat przenośności jest prostszy do zrealizowania w ramach tego systemu, ze względu na uniknięcie bezpośrednich odwołań do sprzętu przez stosowanie standardów (lub de facto standardów), takich jak X-Window, oraz OpenGL lub PEX [18].

Z przedstawionych powyżej względów, jako podstawową platformę sprzętową sugeruje się rodzinę maszyn IRIS firmy Silicon Graphics, pracującą w systemie IRIX (odmiana UNIX) oraz realizującą sprzętowo operacje graficzne, takie jak *z*-bufor i cieniowanie metodą Gouraud. Stacje te są dostępne w wielu wariantach o różnej mocy obliczeniowej, w tym wersji wieloprocesorowej, a także zapewniają binarną kompatybilność oprogramowania.

Struktura funkcjonalna systemu

Współczesne metody syntezy realistycznych obrazów tradycyjnie klasyfikuje się jako należące do jednej z dwóch grup: śledzenia promieni (ang. *ray tracing*) [21] i bilansu promieniowania (ang. *radiosity*) [24]. Podział ten jest w coraz mniejszym stopniu słuszny, zarówno ze względu na powstanie metod hybrydowych, jak i wzajemne wykorzystywanie się metod należących do obu grup (np. zastosowanie śledzenia promieni do wyznaczania współczynników sprzężenia w metodzie bilansu promieniowania [72]). Nie da się jednak zaprzeczyć, że większość zagadnień wywodzi się z jednego z dwóch podejść: dyskretyzacji przestrzeni obrazu (śledzenie promieni) i dyskretyzacji sceny (bilans promieniowania); zakres funkcjonalny systemu do syntezy obrazu musi zatem uwzględniać obie grupy zagadnień.

Jak już wspomniano, najbardziej dojrzałe z zaprezentowanych systemów mają strukturę biblioteczną. Jednocześnie funkcje biblioteczne są pogrupowane w moduły, które z kolei są ułożone w określoną hierarchię. Proponowana architektura ma podobną organizację; co więcej, dozwolony jest tylko dostęp z wyższych poziomów hierarchii do modułów poziomu bieżącego i wszystkich niższych. Każdy moduł powinien mieć własne procedury do prowa-

dzenia statystyki oraz ewentualnie do generowania danych w celach porównawczych. Użytkownicy systemu mogą ingerować w moduły każdego poziomu, dopisując własne procedury: zupełnie nowe, lub funkcjonalnie równoważne już istniejącym; w tym ostatnim przypadku powinny zostać zachowane parametry wywołań (zarówno wejściowe, jak i wyjściowe).

Zaprezentowana poniżej hierarchia systemu odzwierciedla wydzielenie kilku podstawowych komponentów, składających się na opis wizualizowanej sceny: geometrii modelu, atrybutów opisujących materiały, z których model jest zbudowany, oświetlenia i atrybutów ośrodka oraz parametrów widzenia. Poszczególne poziomy hierarchii poczynając od najniższego to:

U. Poziom Modułów Pomocniczych
G. Poziom Geometrii
A. Poziom Atrybutów i Teksturowania
L. Poziom Modelu Oświetlenia
S. Poziom Syntezy Obrazu i Wyświetlania
I. Poziom Obróbki Obrazu
APP. Poziom Aplikacji

U. Poziom Modułów Pomocniczych

Moduły funkcjonalne tego poziomu realizują podstawowe operacje używane na wyższych poziomach, takie jak funkcje matematyczne (np. transformacje, operacje macierzowe i wektorowe, interpolacje, wyznaczanie kierunku promienia odbitego i załamanego), funkcje mieszające i tablice rozproszone, konwersje przestrzeni kolorów i luminancji, dostęp do zbiorów (w tym do zbiorów przechowujących obraz w różnych formatach), dostęp do urządzeń (np. pamięć obrazu, z-bufor, klawiatura, mysz, tablet).

Moduły tego poziomu są niezależne od struktur danych używanych w pozostałej części systemu.

G. Poziom Geometrii

Moduły tego poziomu są odpowiedzialne za operacje na danych geometrycznych opisujących scenę. Jest niezwykle istotne, aby synteza obrazu była niezależna od zewnętrznego opisu geometrycznego sceny, który jest zazwyczaj specyficzny dla konkretnego systemu modelującego. Wymaga to stworzenia uniwersalnej reprezentacji wewnętrznej oraz opracowania konwerterów do powszechnie stosowanych formatów.

Należy podkreślić, że tylko moduły tego poziomu zawierają funkcje dostępu do opisu danych geometrycznych w postaci wewnętrznej i ich obróbki (np. podział wielokątów, tworzenie struktur danych do przyspieszania obliczeń związanego z podziałem przestrzeni). Na tym poziomie znajdują się także funkcje przecięcia obiektów z promieniem, obliczanie wektora normalnego w punkcie, obliczanie współczynników sprzężenia (ang. *form factors*). Większość eksperymentów związana z wykorzystaniem możliwości sprzętu graficznego (oraz dedykowanego, specjalnie konstruowanego sprzętu) jest przeprowadzana właśnie na Poziomie Geometrii.

Ze względu na wielorakość zagadnień (aczkolwiek wszystkie są związane z opisem geometrycznym sceny), Poziom Geometrii został podzielony na kolejne podpoziomy; podział ten, przedstawiony poniżej, jest konsekwencją zastosowania klasyfikacji technik przyspieszających w metodzie śledzenia promieni [77].

G1. Poziom Reprezentacji Wewnętrznej

Moduły tego poziomu są odpowiedzialne za ładowanie i konwersję danych geometrycznych do postaci wewnętrznej. Z wielu przyczyn, powszechnie stosowaną reprezentacją wewnętrzną są wielokąty [68], zatem niezbędne są tu funkcje dokonujące aproksymacji (brył, powierzchni sklejanych i innych form opisu geometrii) za pomocą wielokątów. Z drugiej jednak strony, w modułach wyższych poziomów dostęp do oryginalnych danych może być czasem korzystny, np. do precyzyjnego wyznaczania wektora normalnego lub punktu przecięcia. Niezbędna jest zatem podwójna reprezentacja, co obejmuje także możliwość ładowania wektorów normalnych (patrz opis poziomu G2).

Jak wskazał Baum [5], reprezentacja wielokątowa zazwyczaj wymaga operacji korygujących, zanim będzie ją można zastosować w obliczeniach oświetleniowych. Korekcji geometrii również dokonuje się na poziomie G1.

G2. Poziom Przecięcia Promienia z Obiektem

Na tym poziomie znajdują się moduły odpowiedzialne za znajdowanie punktów przecięcia promienia z obiektami różnych typów oraz do wyznaczania wektorów normalnych. Dodawanie nowych typów obiektów odbywa się przede wszystkim na Poziomie Reprezentacji Wewnętrznej (aproksymacja za pomocą wielokątów), ale może również wymagać uzupełnienia tego poziomu (wyznaczanie punktu przecięcia i wektora normalnego na podstawie reprezentacji zewnętrznej). Tutaj również znajdują się funkcje dostępu do wektorów normalnych (załadowanych poprzez funkcje poziomu G1), stosowane wtedy, gdy uśrednione wektory wielokątów są niewystarczająco dokładne, a analityczne ich obliczenie jest kosztowne (lub niemożliwe, np. ze względu na nieznajomość reprezentacji zewnętrznej).

Na tym poziomie znajdują się także moduły obsługi brył otaczających (ang. *bounding volume*) [11, 74], których celem jest zmniejszenie kosztu przecięcia promienia z obiektem (wybór optymalnego kształtu, przecięcie z bryłą otaczającą). Tutaj przeprowadza się preprocesing obiektów (jednokrotne obliczenie wyrażeń, które mają stałą wartość dla danego obiektu, niezależnie od równania promienia), a także podział wielokątów (reprezentacji wewnętrznej) – operację niezbędną w wielu metodach syntezy obrazu [5, 40].

Należy zwrócić uwagę na możliwość sprzętowej realizacji przynajmniej niektórych z funkcji tego poziomu, np. przecięcia promienia z takimi obiektami, jak wielokąty.

G3. Poziom Podziału Przestrzeni

Zadaniem modułów tego poziomu jest obsługa podziału przestrzeni obiektów w celu redukcji liczby przecięć na drodze promienia, a więc wybór techniki podziału (np. na podstawie zależności statystycznych [66]), a także preprocesing (przypisanie wokselom list obiektów). Jest to miejsce zarówno na zastosowanie znanych algorytmów, takich jak hierarchiczne bryły otaczające [55, 23], octree [20], podział jednorodny [19], BSP [33, 31], podział na regiony [2], jak też na opracowywanie nowych technik oraz algorytmów hybrydowych.

Na tym poziomie najwygodniej jest zrealizować obsługę dużych baz danych, większych niż dostępna pamięć [25]. W tym celu niezbędny jest moduł obsługi pamięci podręcznej (ang. *cache*), przechowującej dane o obiektach z wybranych wokseli (strategia samego wyboru wokseli, czyli sterowania przeładowaniem pamięci cache, znajdzie się na kolejnym, wyższym poziomie). Moduł organizujący manipulację danymi na poziomie woksela umożliwi także realizację algorytmów wieloprocesorowego śledzenia promieni, opartych na podziale przestrzeni obiektów [35, 58]. Również w tym przypadku, strategia dotycząca zorganizowania obliczeń równoległych, znajdzie się na poziomie wyższym.

G4. Poziom Śledzenia Promienia

Jest to poziom, na którym są umieszczone moduły pozwalające na śledzenie pojedynczych promieni. Funkcje obsługi powinny być rozgraniczone ze względu na różną specyfikę poszczególnych rodzajów promieni: pierwotnych, odbitych, załamanych, testujących cienie, wyznaczających współczynniki sprzężenia (ang. *form factors*). Podstawowy moduł śledzenia promienia jest jednak zazwyczaj wspólny i zorganizowany na poziomie woksela. Moduł ten powinien magazynować wszelkie informacje o promieniu i trafionym obiekcie, co pozwoli zrealizować wspomniane równoległe śledzenie promieni (oparte na podziale przestrzeni), sterowanie pamięcią cache, jak też inne zadania, np. „skrzynkę pocztową" promieni [2] lub parametryczne śledzenie promieni [59].

Na Poziomie Śledzenia Promienia wygodne jest też umieszczenie modułów odpowiedzialnych za preprocesing cieni [79, 17, 41].

Szczególna troska przy modularyzacji powinna być podyktowana krytyczną czasochłonnością śledzenia promienia, z którego to powodu poziom ten jest potencjalnym polem do eksperymentów zarówno z wykorzystaniem możliwości typowego sprzętu graficznego [78], jak i sprzętu dedykowanego [46].

A. Poziom Atrybutów i Teksturowania

Moduły tego poziomu odpowiadają za przypisania, bieżącą modyfikację i dostęp do atrybutów obiektów. Każdy obiekt może być opisany dowolną liczbą atrybutów, takich jak kolor, współczynniki odbicia, transmisji, załamania, itd. Atrybuty mogą być wyrażone w postaci tablicowej, uzależnionej od długości fali i kąta padania, co pozwala zrealizować fizycznie uzasadnione modele odbicia [30, 73]. Moduły wyższych poziomów (szczególnie Modelu Oświetlenia) korzystają tylko z tych atrybutów, które są im niezbędne, ignorując pozostałe.

Na tym samym poziomie są umieszczone moduły nakładania (mapowania) tekstur i bumpów [7] oraz tekstury przestrzenne [48,

49], gdyż ich rola polega na modyfikacji atrybutów (ewentualnie wektora normalnego) w punkcie. Wśród modułów mapowania należy wyróżnić moduły filtracji tekstur. Trzeba także zwrócić uwagę na możliwość sprzętowej realizacji algorytmów teksturowania [61].

L. Poziom Modelu Oświetlenia

Na tym poziomie są zgrupowane moduły w ostatecznym rachunku decydujące o jakości generowanego obrazu. Poziom Modelu Oświetlenia jest podzielony na dwa podpoziomy:

L1. Poziom Modelu Odbicia

Główne kategorie zagadnień rozwiązywanych na tym poziomie, to:

- Definiowanie oświetlenia i obsługa różnych typów źródeł światła (wraz z ich przestrzenną dystrybucją promieniowania, tzw. diagramami goniometrycznymi), w tym urojonych świateł o ujemnej energii.
- Wyznaczanie urojonych źródeł światła w celu określenia oddziaływania powierzchni lustrzanych na rozpraszające (metodą luster, ang. image method [71]).
- Strategie próbkowania: źródeł światła, widma.
- Model odbicia, czyli sposób interakcji światła z powierzchnią o atrybutach zdefiniowanych na Poziomie Atrybutów i Teksturowania.

L2. Poziom Wyznaczania Oświetlenia

Produktem wyjściowym podpoziomu L2 (a jednocześnie całego poziomu L) jest określenie oświetlenia i luminancji w punkcie i na powierzchni, zarówno bezpośredniego (od źródeł światła), jak i pośredniego (wpływ oświetlonych powierzchni na rozważany punkt lub powierzchnię). Jako pomocnicze moduły tego poziomu konieczne są strategie próbkowania powierzchni i przestrzeni. Ponadto na poziomie L2 definiuje się model propagacji, określający sposób rozchodzenia się światła w ośrodku, np. tłumienie z odległością, oraz ewentualnie uwzględniający wpływ ośrodków pochłaniających, rozpraszających i emitujących światło [32, 45, 56].

S. Poziom Syntezy Obrazu i Wyświetlania

Poziom ten jest w zasadzie koncepcyjnie bliski Poziomowi Aplikacji i mógłby zostać z nim utożsamiony, jak to uczynił Trumbore [68]. Jednakże rozdzielenie tych dwóch poziomów wydaje się pożyteczne, gdyż pozwala traktować poziom S jako narzędzia do realizacji kompletnego modelu oświetlenia i generowania obrazu, czyli swego rodzaju nakładkę na poziom L, który realizuje lokalny (i częściowo globalny) model oświetlenia dla punktu i powierzchni. Istotną przesłanką jest tutaj wynikająca z czasochłonności syntezy obrazu tendencja do konstruowania algorytmów progresywnych [13, 89], które umożliwiają szybsze przybliżenie ostatecznych rezultatów przez zastosowanie wyników częściowych. Czas osiągnięcia rozwiązania końcowego pozostaje ten sam (lub nawet nieznacznie się zwiększa), ale progresywne, krokowe poprawianie dokładności daje możliwość bieżącej oceny i interwencji użytkownika przed zakończeniem obliczeń.

Moduły syntezy obrazu realizują zatem:

- Definiowanie parametrów syntetycznej kamery i ścieżki animacyjnej.
- Strategie próbkowania przestrzeni obrazu, np. wykorzystujące jego spójność [1], czy diagonalny wzorzec próbkowania [9].
- Strategie niwelowania zjawiska aliasingu, np. próbkowanie stochastyczne [39, 47, 60].
- Rozproszone śledzenie promieni [14].
- Sterowanie parametrycznym śledzeniem promieni [59].
- Krokowe, adaptacyjne poprawianie dokładności obrazu [6].
- Zarządzanie statycznym i dynamicznym równoważeniem obciążenia w przetwarzaniu równoległym [36, 58, 52, 76].
- Iteracyjne wyznaczanie wzajemnego oświetlania powierzchni [13, 40].
- Oszacowanie oświetlenia, które zostanie doliczone w następnych iteracjach [13].
- Strategia podejmowania decyzji o adaptacyjnym podziale geometrii (wykonywanym na poziomie G2) na podstawie obliczeń oświetleniowych (wykonywanych na poziomie L2) [12, 70, 78].
- Techniki inkrementalnego generowania obrazów dla sekwencji animowanych i stereowizji [4, 26].
- Monitorowanie i rejestrowanie interakcji [29].

Moduły związane z wyświetlaniem są odpowiedzialne za takie zagadnienia, jak:

- Reprodukcja koloru [28, 37].
- Dopasowanie obliczonej luminancji do zakresu luminancji monitora [38] i korekcja nieliniowości lampy monitora (korekcja gamma).
- Uwzględnienie psychofizjologicznych aspektów widzenia.
- Superpozycja obrazu wygenerowanego sprzętowo z wynikami obliczeń uzupełniających [78].

I. Poziom Obróbki Obrazu

Moduły funkcjonalne tego poziomu operują na wygenerowanym obrazie, realizując takie efekty jak filtracja, głębia ostrości [50], mgła, lub raczej perspektywa powietrzna [57], aureole wokół źródeł światła [44], a także komponowanie obrazów w tym z obrazami ze skanera [45].

APP. Poziom Aplikacji

Na szczycie hierarchii znajdują się aplikacje, zbudowane z modułów wszystkich niższych poziomów, będące w istocie kompletnymi metodami prowadzącymi do wygenerowania obrazu oraz danych liczbowych (w tym danych do porównań). Zbudowanie amplikacji, to w istocie dokonanie wyboru i powiązanie modułów, po ewentualnym uzupełnieniu ich na każdym poziomie (na którym jest to konieczne) o nowo opracowane moduły. Aplikacjami w tym sensie są np. metoda śledzenia promieni oraz metoda bilansu promieniowania, metoda dwukierunkowego śledzenia promieni [40], a także metody hybrydowe [71, 10].

Zaprezentowana propozycja architektury otwartego systemu do syntezy obrazów realistycznych jest wynikiem doświadczeń, nabytych przez autora podczas projektowania implementacji i użytkowania systemu symulacji oświetlenia i syntezy obrazu Turbo Beam Tracing (TBT) japońskiej firmy Integra Inc., edukacyjnego systemu syntezy obrazu GrafSYN (realizowanego w ramach programu resortowego RRI.14), systemu syntezy obrazu ARETE firmy ARCONEX Ltd., a także użytkowania systemów RayShade i RADIANCE.

Należy stwierdzić, że systemy dydaktyczno-badawcze rzadko są tworzone od podstaw; zadanie polegające na zbudowaniu systemu w pełni odpowiadającego podanym założeniom musi być kosztowne i czasochłonne. Z drugiej strony, nie jest to zazwyczaj konieczne: większość ośrodków akademickich i badawczych ma konkretne obszary zainteresowań. W tych warunkach, rozsądnym wydaje się potraktowanie wybranego pakietu ogólnodostępnego (*public domain*) jako bazy do stworzenia systemu zgodnego z przedstawionymi założeniami. Naturalnymi kandydatami są w obecnej chwili wspomniane systemy RADIANCE i RayShade, dostępne (co jest niezbędnym warunkiem) w postaci źródłowej. Pomimo zaliczenia ich do grupy systemów monolitycznych, ich architektura jest z przyczyn projektowych modularna, a chociaż dokumentacja techniczna jest dość uboga, zapowiadane jest jej rozszerzenie. Istotny jest także fakt, że systemy te są przekształcane i rozwijane w wielu ośrodkach akademickich, a wyniki prac (w tym kod źródłowy i dane) są często udostępniane w postaci biuletynów i publikacji. Prawdopodobnie za najbardziej obiecujący należałoby uznać system SPECTRUM, który stawia sobie cele podobne do zaprezentowanych.

Autor składa podziękowanie Karolowi Myszkowskiemu za cenne uwagi, które przyczyniły się do nadania prezentowanemu artykułowi ostatecznego kształtu.

LITERATURA

[1] Akimoto T., Mase K., Suenaga Y.: Pixel-selected ray tracing. IEEE, Comput. Graph. Appl. 11 (4): 14–22, July 1991

[2] Arnaldi B., Priol T., Bauatouch K.: A new space subdivision method for ray tracing CSG modelled scenes. The Visual Computer 3, 1987 pp. 98–108

[3] Arvo J.: Ray Tracing with Meta-Hierarchies, in: Advanced Topics in Ray Tracing. SIGGRAPH 1990 Course Notes, Vol. 24

[4] Badt S.: Two algorithms for taking advantage of temporal coherence in ray tracing. The Visual Computer 4, 1988, pp. 123–132

[5] Baum D., Mann S., Smith K., Winget J.: Making Radiosity Usable: Automatic Preprocessing and Meshing Techniques for the Generation of Accurate Radiosity Solutions. ACM Comput. Graph. 25 (4) 1988, pp. 51–60

[6] Bergman L., Fuchs H., Grant E.: Image Rendering by Adaptive Refinement. ACM Comput. Graph., 20 (4) 1986, pp. 29–37

[7] Blinn J.: Simulation of wrinkled surfaces. ACM Comput. Graph. (Siggraph'78 Proceedings: pp. 286–292)

[8] Blinn J.: System Aspects of Computer Image Synthesis and Computer Animation. SIGGRAPH 1984 Advanced Image Synthesis Course Notes

[9] Bouville C., Tellier P., Bauatouch K.: Low sampling densities using a psychovisual approach. Proc. of Eurographics'91, pp. 167–182
[10] Chen E.S., Rushmeier H.E., Miller G., Turner D.: A progressive multi-pass method for global illumination. ACM Comput. Graph., 25 (4) 1991, pp. 165–174
[11] Clark J.H.: Hierarchical geometric models for visible surface algorithms. Comm ACM 19 (10) 1976, pp. 547–554
[12] Cohen M., Greenberg D., Immel S., Brock P.: An Efficient Radiosity Approach for Realistic Image Synthesis. IEEE Comput. Graph. Appl., 6 (2) 1986, pp. 26–35
[13] Cohen M., Chen S., Wallace J., Greenberg D.: A Progressive Refinement Approach to Fast Radiosity Image Generation. ACM Comput. Graph. 22 (4) 1988, pp. 75–84
[14] Cook R.L., Porter T., Carpenter L.: Distributed ray tracing. ACM Comput. Graph., 18 (4) 1984, pp. 165–174
[15] Cook R., Carpenter L. Catmull E.: The REYES Image Rendering Architecture. ACM Comput. Graph., 21 (4) 1987, pp. 95–102
[16] Crow F.: A More Flexible Image Generation Environment. ACM Comput. Graph. 16 (3) 1982, pp. 9–18
[17] Eo D., Kyung C.: Hybrid shadow testing scheme for ray tracing. Computer Aided Design, 21 (1) 1989, pp. 38–48
[18] Foley J.: 3D Graphics Standards Debate: PEX vs. OpenGL, Panel Session, J. Foley chair. ACM Comput. Graph., 26 (2) 1992, pp. 408–409
[19] Fujimoto A., Tanaka T., Iwata K.: ARTS: Accelerated Ray-Tracing System. IEEE Comput. Graph. Appl. 6 (4), 1986, pp. 16–26
[20] Glassner A.: Space subdivision for fast ray tracing. IEEE Comput. Graph. Appl., 4 (10) 1984, pp. 15–22
[21] Glassner A.: An Introduction to Ray Tracing. Academic Press, 1989
[22] Glassner A.: Spectrum – A Proposed Image Synthesis Architecture, in: Eurographics 1991 Tutorial Note. The Theory and Practice of Ray Tracing, pp. 33–135
[23] Goldsmith J., Salmon J.: Automatic creation of object hierarchies for ray tracing. IEE Comput. Graph. Appl., 7 (5) 1987, pp. 14–20
[24] Greenberg D., Cohen M., Torrance K.: Radiosity: A Method for Computing Global Illumination. The Visual Computer, 2 (5) 1986, pp. 291–297
[25] Green S.A., Paddon D.J.: A highly flexible multiprocessor solution for ray tracing. The Visual Computer 6, 1990, pp. 62–73
[26] Groller E., Purgathofer W.: Using temporal and spatial coherence for accelerating the calculation of animation sequences. Proc. of Eurographics'91, pp. 103–113
[27] Hall R., Greenberg D.P.: A testbed for realistic image synthesis. IEEE Comput. Graph. Appl., 3 (10) 1983, pp. 10–20
[28] Hall R.: Color reproduction and illumination models. Techniques for Computer Graphics, eds. Rogers and Earnshaw, pp. 194–238
[29] Hall R., Bussan M., Georgiades P., Greenberg D.: A Testbed for Architectural Modeling. Eurographics 1991 Conference Proceedings, North-Holland 1991, pp. 47–58
[30] He X.D., Torrance K.E., Sillion F., Greenberg D.P.: A comprehensive physical model for light reflection. ACM Comput. Graph., 25 (4) 1991, pp. 175–186
[31] Jansen F.: Data structures for ray tracing. Eurographics seminar Data Structures for Raster Graphics, Kessener, Peters, van Lierop eds, Springer-Verlag, pp. 57–73
[32] Kajiya J.T., von Herzen B.P.: Ray tracing volume densities. ACM Comput. Graph., 18 (3) 1984, pp. 165–174
[33] Kaplan M.R.: Space-tracing, a constant time ray tracer. SIGGRAPH'85 State of the Art in Image Synthesis, seminar notes
[34] Kirk D., Arvo J.: The ray tracing kernel. Proc. of Ausgraph'88, Melbourne, Australia, 1988, pp. 75–82
[35] Kobayashi H., Nakamura T., Shigei Y.: Parallel processing of an object space for image synthesis using ray tracing. The Visual Computer 3, 1987, pp. 13–22
[36] Kobayashi H., Nishimura S., Kubota H., Nakamura T., Shigei Y.: Load balancing strategies for a parallel ray-tracing system based on constant subdivision. The Visual Computer 4, 1988, pp. 197–209
[37] Mayer G.: Wavelength selection for synthetic image generation. Computer Vision, Graphics and Image Processing, vol. 41, 1988, pp. 57–79
[38] Miller N.J., Ngai P.Y., Miller D.D.: The application of computer graphics in lighting design. Journal of Illuminating Engineering Society, 14 (October) 1984, pp. 6–26
[39] Mitchell D.P.: Generating antialiased images at los sampling densities. ACM Comput. Graph., 21 (4), 1987, pp. 65–72
[40] Myszkowski K.: Synteza realistycznych obrazów metodą dwukierunkowego śledzenia promieni. Praca doktorska, Wydział Elektroniki Politechniki Warszawskiej
[41] Myszkowski K., Wojdała A.: Zastosowanie metod syntezy obrazu do symulacji oświetlenia. Archiwum Informatyki Teoretycznej i Stosowanej, t. 1, z. 1–4, 1989, s. 119–133
[42] Nadas T., Fournier A.: GRAPE: An Environment to Build Display Processes. ACM Comput. Graph., 21 (4) 1987, pp. 75–84
[43] Nakamae E., Harada K., Ishizaki T., Nishita T.: A montage method: the overlaying of the computer gennerated images onto a background photograph. ACM Comput. Graph., 20 (4) 1986, pp. 207–214
[44] Nakamae E., Kaneda K., Okamoto T., Nishita T.: A lighting model aiming at drive simulators. ACM Comput. Graph., 24 (4) 1990, pp. 395–404
[45] Nashita T., Miyawaki Y., Nakamae E.: A shading model for atmospheric scattering considering lumnous intensity distribution of light sources. ACM Comput. Graph., 21 (4) 1987, pp. 303–310
[46] Olech B.: Sprzętowe wspomaganie algorytmów śledzenia promieni. Zgłoszone do publikacji w: Informatics & Automatics, Politechnika Szczecińska, 1993
[47] Painter J., Sloan K.: Antialiased ray tracing by adaptive progressive refinement. ACM Comput. Graph., 23 (3) 1989, pp. 281–288
[48] Peachey D.R.: Solid texturing of complex surfaces. ACM Comput. Graph., 19 (3) 1985, pp. 279–286
[49] Perlin K.: An image synthesizer. ACM Comput. Graph., 19 (3) 1985, pp. 287–296
[50] Potmesil M., Chakravarty I.: A lens and aperture camera model for synthetic image generation. ACM Comput. Graph., 15 (3) 1981, pp. 297–305
[51] Potmesil M., Hoffert E.: FRAMES: Software Tools for Modeling, Rendering and Animation of 3D Scenes. ACM Comput. Graph. 21 (4) 1987, pp. 85–94
[52] Priol T., Bauatouch K.: Static load balancing for a parallel ray tracing on a MIMD hypercube. The Visual Computer 5, 1989, pp. 109–119
[53] RADIANCE: User Documentation. Lawrence Berkeley Laboratory, California, 1992
[54] RayShade: User Documentation for version 4.0.6, 1991
[55] Rubin S.M., Whitted T.: A 3-dimensional representation for fast rendering of complex scenes. ACM Comput. Graph., 14 (3) 1980, pp. 110–116
[56] Rushmeier H.E., Torrance K.E.: The zonal method for calculating light intensities in the presence of a participating medium. ACM Comput. Graph., 21 (4) 1987, pp. 293–302
[57] Schachter B.: Generation of special effects. Computer Image Generation, John Wiley, New York, pp. 155–172
[58] Scherson I.D., Caspary E.: Multiprocessing for ray tracing: a hierarchical self-balancing approach. The Visual Computer, 4, 1988, pp. 188–196
[59] Sequin C.H., Smyrl E.K.: Parametrized ray tracing. ACM Comput. Graph., 23 (3) 1989, pp. 307–314
[60] Shirley P.: Discrepancy as a quality measure for sample distributions. Proc. of Eurographics'91, pp. 183–194
[61] Silicon Graphics: IRIS Graphics Library Programming Guide. Silicon Graphics, Mountain View, California, 1990
[62] Silicon Graphics: IRIS Explorer Manuals (User's Guide, Module Definitions, Module Writer's Guide). Silicon Graphics, Mountain View, California 1991
[63] Silicon Graphics: IRIS Performer Programming Guide Silicon Graphics, Mountain View, California, 1992
[64] Strauss P.: BAGS: The Brown Animation Generation System. Tech. Report CS 88-27. Dept. of Comp. Science, Brown. Univ., Providence, Rhode Island, 1988
[65] Strauss P., Carey R.: An Object-Oriented 3D Graphics Toolkit. ACM Comput. Graph., 26 (2) 1992, pp. 341–349
[66] Sung K.: A DDA Octree Traversal Algorithm for Ray Tracing. Eurographics'91 Proceedings, pp. 73–85
[67] Magnenat-Thalmann N., Thalmann D.: MIRALab image synthesis software, in: Image Synthesis: Theory and Practice, Springer-Verlag, Tokyo 1987
[68] Trumbore E., Lytle W., Greenberg D.: A Testbed for Image Synthesis. Eurographics 1991 Conference Proceedings, North-Holland 1991, pp. 467–480
[69] Upstill S.: The RenderMan Companion. A Programmer's Guide to Realistic Computer Graphics. Addison-Wesley 1990
[70] Vedel C., Puech C.: A testbed for adaptive subdivision in progressive radiosity. Proceedings of Second Eurographics Workshop on Rendering, Barcelona 1991
[71] Wallace J.R., Cohen M.F., Greenberg D.P.: A Two-pass solution to the rendering equation: A synthesis of Ray Tracing and Radiosity Methods. ACM Comput. Graph., 21 (4) 1987, pp. 311–320
[72] Wallace J.R., Elmquist K.A. Haines E.A.: A ray tracing algorithm for progressive radiosity. ACM Comput. Graph., 23 (4) 1989, pp. 315–324
[73] Ward G.: Measuring and Modeling Anisotropic Reflection. ACM Comput. Graph., 26 (2) 1992, pp. 265–272
[74] Whitted T.: An improved illumination model for shaded display. Commun. ACM, 23 (6) 1980, pp. 343–349

dokończenie na s. 28

ARIS – Architektura Zintegrowanych Systemów Informacyjnych

A.-W. Scheer, T. Elsner
Institut für Wirtschaftsinformatik
Universität des Saarlandes
D. Dziuba, M. Lasek
Wydział Nauk Ekonomicznych
Uniwersytet Warszawski

ARIS (niem. *Architektur Integrierter Informationssysteme*, ang. *Architecture of Integrated Information Systems*) jest podejściem metodycznym i zestawem programów dla modelowania i realizacji przedsięwzięć informatycznych. Podstawy teoretyczne ARIS zostały opracowane przez prof. A.-W. Scheera z Uniwersytetu Saarbrücken. Zostały one przedstawione w jego pracy: *Architektur Integrierter Informationssysteme*, wydanej przez Springer-Verlag w 1991 r. [8]. Angielskojęzyczna wersja tej pracy ukazała się w 1992 r. pod tytułem: *Architecture of Integrated Information Systems*, także nakładem Springer-Verlag.

Użyte w nazwie słowo architektura, jest odwołaniem się do architektury w budownictwie – sztuki budowania, która od tysięcy lat dostarcza reguł planowania i realizacji projektów budowlanych. Według tych reguł muszą pracować wszyscy biorący udział w projekcie partnerzy (rzemieślnicy, firmy budowlane itp.). Podobnie jak w przypadku projektów budowlanych, w realizacji przedsięwzięć informatycznych bierze udział wielu partnerów (kierownictwo przedsiębiorstwa, informatycy, ekonomiści, zewnętrzni eksperci). Aby zrealizowany przez nich system informatyczny spełniał później wymagania spójności technicznej i organizacyjnej, niezbędna jest praca wszystkich partnerów według jednakowych reguł. Propozycję takich reguł stanowi ARIS. Reguły te określają komponenty, jakie tworzą system informacji („co ma być budowane?"), oraz sposób postępowania (procedurę) budowy systemu („jak ma być budowane?").

Punktem wyjścia dla realizacji systemu informatycznego jest analiza łańcuchów procesów w informatyzowanej organizacji. Pod pojęciem procesów rozumie się pracochłonne działania, rozpoczynające i kończące się określonymi zdarzeniami. Zdarzenie początkowe i końcowe definiują początek i koniec procesu. W niektórych przypadkach proces może zainicjować i kończyć wiele zdarzeń.

Przykładami łańcuchów procesów są:

- zarządzanie zleceniem w przedsiębiorstwie przemysłowym, począwszy od akceptacji zlecenia, przez zagospodarowanie materiałów i produkcję, aż do wysyłki wytworzonych produktów,
- realizacja nowego wyrobu, począwszy od jego koncepcji, przez projektowanie i testowanie prototypu, aż do skierowania przetestowanego produktu do produkcji.

W procesach mamy do czynienia zarówno z przetwarzaniem materialnym, jak i przetwarzaniem informacji. W zależności od rodzaju procesu dominować może przetwarzanie materialne lub informacji. W procesach produkcji przemysłowej dominuje przetwarzanie materialne, natomiast w procesach planowania i projektowania – proces przetwarzania informacji. Przetwarzanie materialne i informacji przeplatają się wzajemnie. Przetwarzanie materialne pociąga za sobą przetwarzanie informacji, przetwarzanie informacji może dać początek przetwarzaniu materiałowemu. Te powiązania są bardzo dobrze widoczne w przypadku produkcji, w której stosuje się urządzenia sterowane numerycznie.

Łańcuchy procesów są rozpatrywane na trzech różnych poziomach abstrakcji, co przedstawiono na rys. 1 ilustrującym przykład łańcuchów procesów dla realizacji zleceń produkcyjnych. Pierwszy, najniższy poziom tworzą opisy łańcuchów procesów występujących w rzeczywistości, a więc obejmujących realne obiekty. Drugi, wyższy poziom, to opis łańcuchów procesów, w którym operuje się klasami obiektów, występujących na najniższym poziomie. Wreszcie trzeci poziom stanowi tzw. meta-poziom, to jest poziom opisu dla opisów łańcuchów procesów. Wyodrębnianie klas obiektów na poszczególnych poziomach jest po części subiektywne i dlatego możliwe są różne hierarchie klas obiektów.

Analizując łańcuchy procesów mamy do czynienia z dużą liczbą elementów, które muszą być ujęte w opisie. Są to: warunki, zdarzenia, procesy, materiały, ludzie i sprzęt (produkcyjny i informatyczny), jednostki organizacyjne. Biorąc pod uwagę wzajemne powiązania tworzy to bardzo złożoną strukturę. Aby tę złożoność zredukować:

- ujmuje się jako część środowiska (tzw. warunki środowiskowe) te elementy, które bezpośrednio nie stanowią przetwarzania informacji,
- prowadzi się odrębne opisy z różnych **punktów widzenia**,
- rozbija się na kolejno realizowane fazy proces realizacji przedsięwzięcia informatycznego.

Przykładem elementu, który nie jest traktowany jako odrębna część, lecz jako część środowiska, jest przedstawiony na rys. 2 proces produkcji materialnej. Nie znaczy to, że informacje o produkcji materialnej zostają zagubione. Są one elementem składnika **warunki (uwarunkowania) środowiskowe.**

Opis łańcuchów procesów jest obciążony znaczną redundancją informacji. Te same warunki środowiskowe, zdarzenia, użytkownicy itp. dotyczą wielu procesów. Aby pozbyć się tej redundancji, łańcuch procesów jest rozpatrywany z różnych **punktów widzenia.** Koncepcję tę ilustruje rys. 3, na którym wyróżniono następujące elementy:

- dane,
- procesy-funkcje,
- organizację,
- środki techniki informatycznej.

Dla uwzględnienia w opisie wzajemnych powiązań wprowadza się dodatkowy element, określany jako sterowanie. Rezygnuje się natomiast z odrębnego traktowania środków techniki informatycznej, rozpatrując je w ramach innych elementów. Opisy z różnych **punktów widzenia** są traktowane jako niezależne całości. Oznacza to, że podczas realizacji systemu informatycznego każdy **punkt widzenia** może być przedmiotem oddzielnego opisu.

Przyjmuje się następujący pięciofazowy cykl życia systemu informatycznego:

Faza 1. Niesformalizowany opis w języku użytkownika. Jest on opracowywany na podstawie analizy istniejących łańcuchów procesów. Nie prowadzi się odrębnych opisów z różnych **punktów widzenia,** lecz są one tu rozpatrywane łącznie.

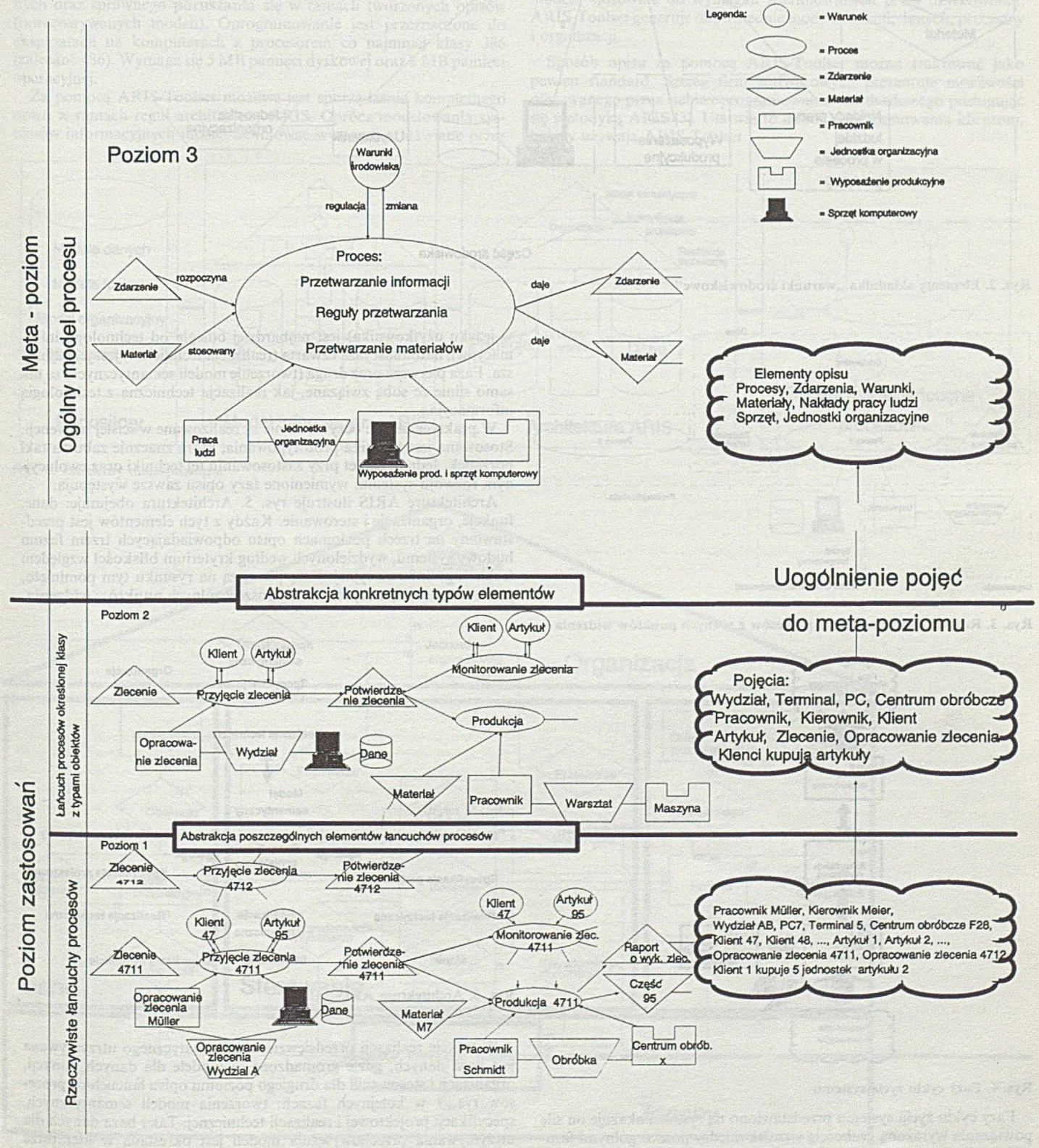
Faza 2. Tworzenie modeli semantycznych. Za pomocą tych modeli opisuje się w sposób sformalizowany koncepcje z fazy pierwszej. Problem jest już w tej fazie, podobnie jak w każdej następnej fazie, rozpatrywany oddzielnie dla różnych **punktów widzenia.** Jest to opis bliski użytkownika, lecz w postaci na tyle sformalizowanej, aby stanowić punkt wyjścia dla projektowania fazy następnej. Nie rozważa się problemów związanych ze stosowaniem technik informatycznych.

Faza 3. Specyfikacja projektowa. W fazie tej uwzględnia się aspekty techniczne: techniki informacyjne i komunikacyjne, takie jak stosowanie systemów baz danych, sieci, języków programowania. Opis nie odnosi się jednak do konkretnych, dostępnych na rynku informatycznym produktów.

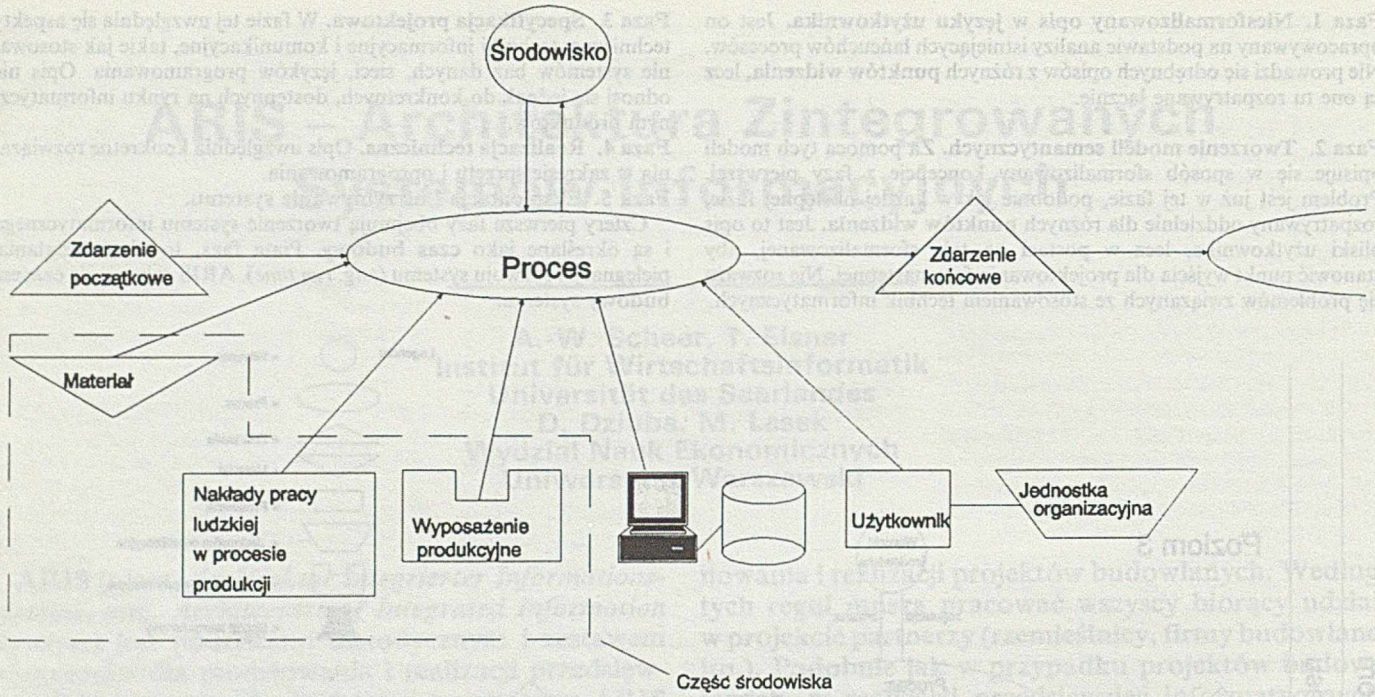
Faza 4. Realizacja techniczna. Opis uwzględnia konkretne rozwiązania w zakresie sprzętu i oprogramowania.

Faza 5. Eksploatacja i utrzymywanie systemu.

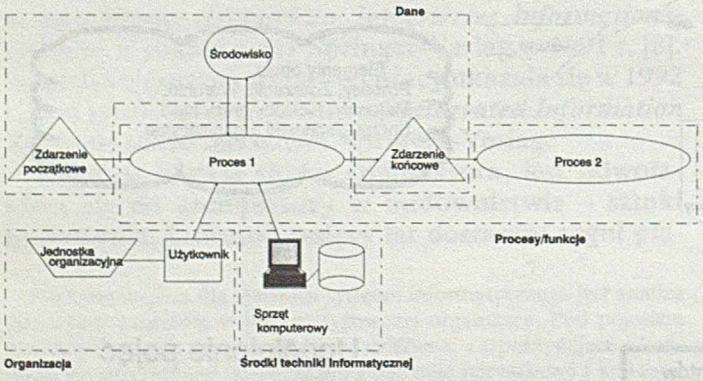
Cztery pierwsze fazy obejmują tworzenie systemu informatycznego i są określane jako **czas budowy.** Piąta faza, to okres działania, pielęgnacji i rozwoju systemu (ang. *run time*). ARIS zajmuje się **czasem budowy** systemu.



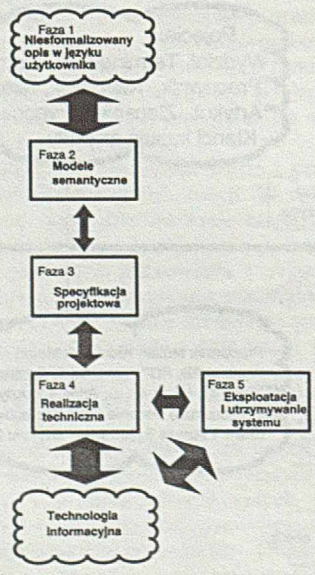
Rys. 1. Poziomy abstrakcji rozpatrywania łańcuchów procesów



Rys. 2. Elementy składnika „warunki środowiskowe"



Rys. 3. Rozpatrywanie łańcucha procesów z różnych punktów widzenia

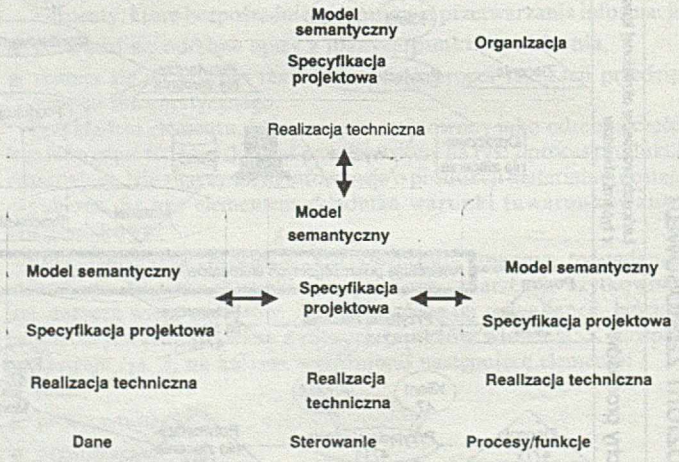


Rys. 4. Fazy cyklu życia systemu

Fazy cyklu życia systemu przedstawiono na rys. 4. Pokazuje on siłę powiązania wyrażoną grubością strzałek między poszczególnymi fazami oraz między fazami a technologią informacyjną. Faza pierwsza (opis w języku użytkownika) jest najbardziej odległa od technologii informacyjnej, natomiast faza czwarta (realizacja techniczna) jest jej najbliższa. Faza pierwsza oraz druga (tworzenie modeli semantycznych) są tak samo silnie ze sobą związane, jak realizacja techniczna z technologią informacyjną.

W praktyce fazy te oczywiście nie są realizowane w ścisłej sekwencji. Stosowana jest technika prototypowania, która znacznie zaburza taki porządek. Jednak nawet przy zastosowaniu tej techniki oraz ewolucyjnym rozwoju systemu, wymienione fazy opisu zawsze występują.

Architekturę ARIS ilustruje rys. 5. Architektura obejmuje: dane, funkcje, organizację i sterowanie. Każdy z tych elementów jest przedstawiany na trzech poziomach opisu odpowiadających trzem fazom budowy systemu, wydzielonych według kryterium **bliskości** względem technologii informacyjnej. Fazę pierwszą na rysunku tym pominięto, ponieważ nie rozpatruje się w niej poszczególnych **punktów widzenia.**



Rys. 5. Architektura ARIS

W trakcie realizacji przedsięwzięcia informatycznego utrzymywana jest baza danych, gdzie gromadzone są modele dla danych, funkcji, organizacji i sterowania dla drugiego poziomu opisu łańcuchów procesów rys. 1 w kolejnych fazach: tworzenia modeli semantycznych, specyfikacji projektowej i realizacji technicznej. Taka baza danych dla utrzymywania przechowywania modeli jest określana w literaturze informatycznej mianem składnicy informacji (ang. *repository*) [11].

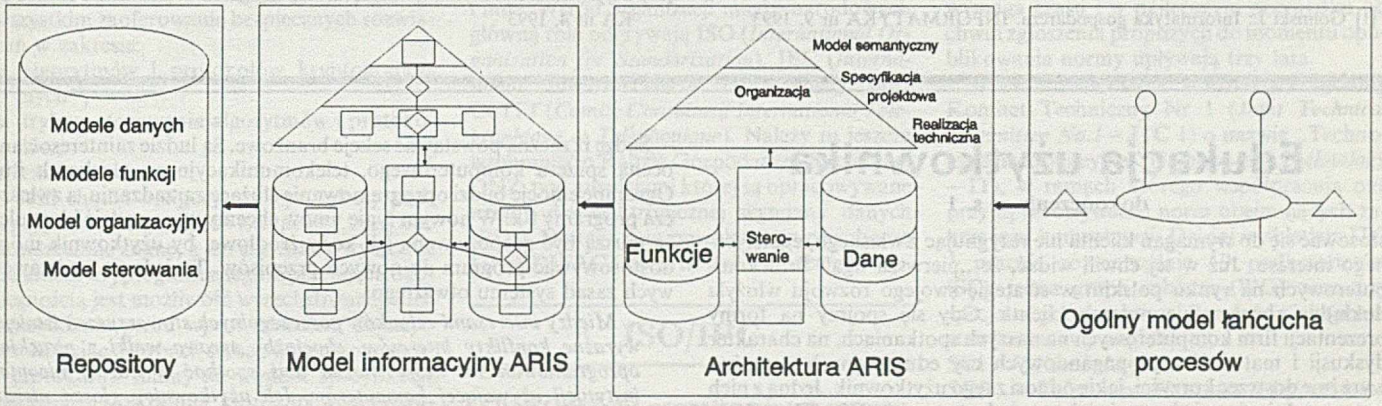
Schemat pojęciowy takiej bazy danych jest zdefiniowany za pomocą modelu meta-informacji. W ARIS dla opisu na poziomie modelu meta-informacji (trzeci poziom z rysunku 1) stosuj się model ERM Chena (*Chen's Entity-Relationship Model*), ponieważ pozwala on dobrze przedstawiać obiekty i relacje między nimi. Składniki meta-poziomu przedstawiono na rys. 6. W pracy [8] przedstawiono szczegółowo model meta-informacji architektury ARIS (rys. 7).

W firmie IDS prof. Scheer GmbH opracowano oprogramowanie o nazwie ARIS-Toolset, którego zadaniem jest wspomaganie realizacji przedsięwzięć informatycznych zgodnie z podejściem metodycznym ARIS. Oprogramowanie to jest przeznaczone do analizy istniejącego stanu przetwarzania informacji, modelowania systemów informacyjnych oraz sprawnego **poruszania się** w ramach tworzonych opisów (opracowywanych modeli). Oprogramowanie jest przeznaczone do eksploatacji na komputerach z procesorem co najmniej klasy 386 (zalecane 486). Wymaga się 5 MB pamięci dyskowej oraz 8 MB pamięci operacyjnej.

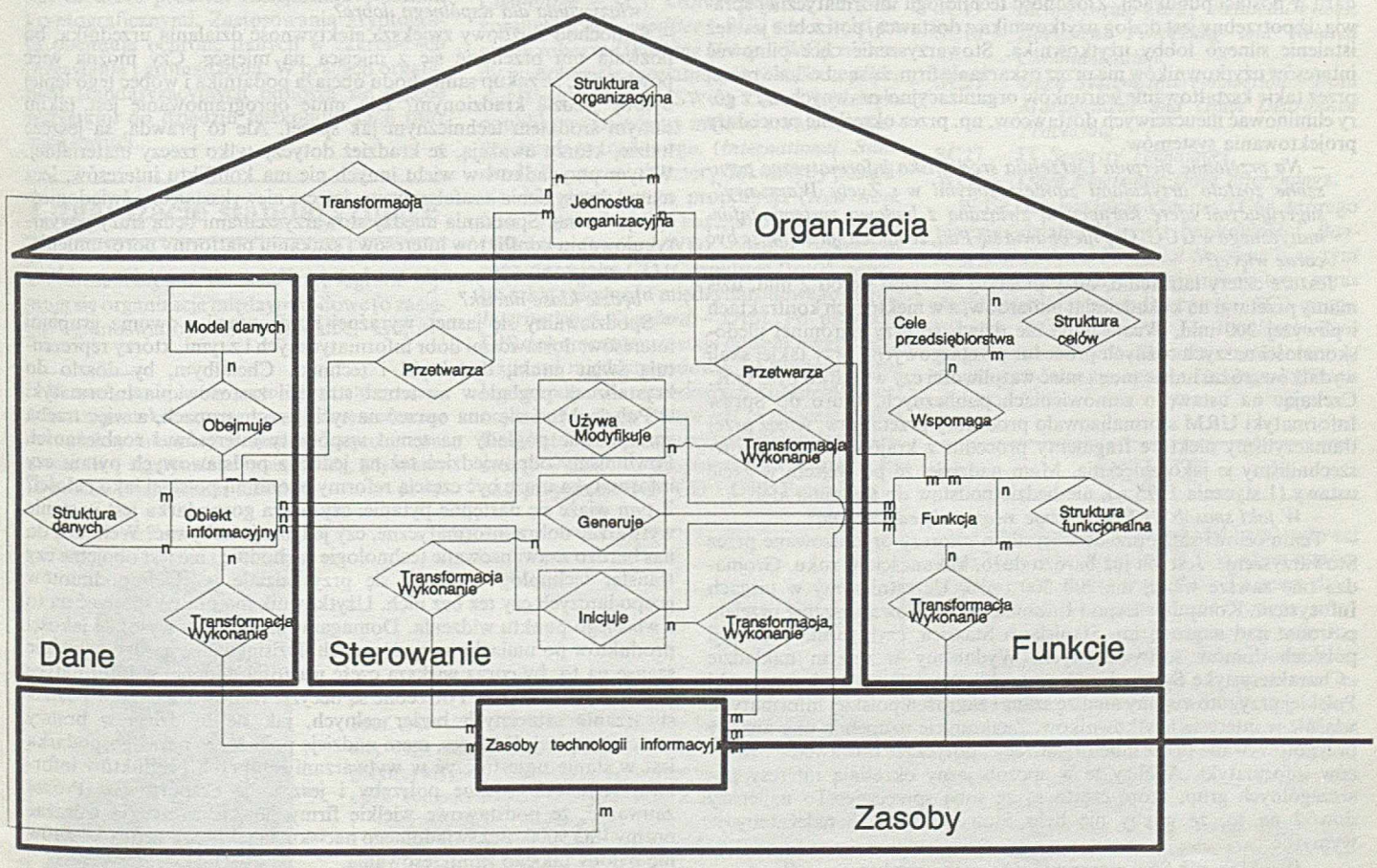
Za pomocą ARIS-Toolset możliwe jest sporządzenie kompletnego opisu w ramach reguł architektury ARIS. Oprócz modelowania systemów informacyjnych można definiować wymagania stawiane przez przedsiębiorstwo oprogramowaniu standardowemu. Do opisu tych wymagań wykorzystuje się schematy funkcjonalne (drzewa funkcji), struktur danych, procesów oraz schematy organizacyjne. W razie potrzeby sporządzone opisy mogą stanowić podstawę do tworzenia własnego oprogramowania. Sposób modelowania funkcji, danych, procesów i organizacji we wczesnych fazach tworzenia systemu zapewnia dużą elastyczność realizacji technicznej, a więc rozwiązań w zakresie sprzętu i oprogramowania.

Opierając się na architekturze ARIS opracowano w firmie IDS Prof. Scheer GmbH szereg modeli referencyjnych, odnoszących się do poszczególnych branż przemysłowych [2, 4, 9, 10]. Na podstawie tych modeli, stosownie do wymagań zdefiniowanych przez użytkownika, ARIS-Toolset generuje odpowiednie modele funkcji, danych, procesów i organizacji.

Sposób opisu za pomocą ARIS-Toolset można traktować jako pewien standard. Szereg firm softwarowych prezentuje możliwości oferowanego przez siebie oprogramowania standardowego posługując się metodyką ARIS [3]. Ułatwia to dobór oprogramowania klientom, którzy używają ARIS-Toolset.



Rys. 6. Elementy meta-poziomu



Rys. 7. Model meta-informacji architektury ARIS

Oprogramowanie ARIS-Toolset jest oferowane w trzech wersjach językowych: niemieckiej, angielskiej, francuskiej. Do głównych nabywców tego oprogramowania, a także i modeli referencyjnych należą firmy konsultingowe, średnie i duże przedsiębiorstwa (koncerny) przemysłowe różnych branż oraz firmy ubezpieczeniowe, banki i przedsiębiorstwa turystyczne.

Architektura ARIS umożliwia pokazanie sposobu rozpatrywania i analizowania systemów informacyjnych przez ekonomistów, pozwalającego realizować ich własne koncepcje. Powiązaniami i wzajemnymi oddziaływaniami między ekonomią a informatyką zajmuje się informatyka gospodarcza [1, 5, 6, 7, 9). Wprowadzenie kształcenia w zakresie informatyki gospodarczej na polskich uniwersytetach jest jednym z celów realizowanego przez uniwersytety w Saarbrücken, Nancy oraz Warszawie projektu w ramach programu TEMPUS, finansowanego przez Europejską Wspólnotę Gospodarczą. W ramach tego projektu opracowywana jest koncepcja oraz program studiów z zakresu informatyki gospodarczej. Proponuje się m.in. nauczanie tworzenia, optymalizacji oraz technicznej realizacji zintegrowanych informatycznych systemów użytkowych wykorzystując architekturę ARIS.

LITERATURA

[1] Goliński J.: Informatyka gospodarcza. INFORMATYKA nr 9, 1993

[2] Hars A.: Referenzdatenmodelle, Grundlagen effizienter Datenmodellierung. Gabler Verlag, Wiesbaden 1994

[3] Keller G., Meinherdt S.: SAP R/3 – Analyzer. Walldorf 1994

[4] Mattheis P.: Prozessorientierte Informations- und Organisationsstrategie, Analyse, Konzeption, Realisierung. Gabler Verlag, Wiesbaden 1993

[5] Mertens P.: Wirtschaftsinformatik, Studien und Forschungsführer. Springer-Verlag 1992

[6] Scheer A.-W.: EDV-orientierte Betriebswirtschaftslehre. Springer-Verlag 1990

[7] Scheer A.-W.: CIM Computer Integrated Manufacturing – Der computergesteuerte Industriebetrieb. Springer-Verlag 1990

[8] Scheer A.-W.: Architektur integrierter Informationssysteme. Springer-Verlag 1991 (wersja w języku angielskim: Scheer A.-W.: Architecture of Integrated Information Systems. Springer-Verlag 1992)

[9] Scheer A.-W.: Wirtschaftsinformatik, Referenzmodelle für industrielle Geschäftsprozesse. Springer-Verlag 1994

[10] Schwarzer B.: Prozessorientiertes Informationsmanagament in multinationalen Unternehmen. Eine empirische Untersuchung in der Pharmaindustrie. Gabler Verlag, Wiesbaden 1994

[11] Szyjewski Z.: Klasyfikacja narzędzi wspomagania CASE. INFORMATYKA nr 4, 1993

Edukacja użytkownika

dokończenie ze s. 1

stosować się do wymagań klienta nie rezygnując z własnego, egoistycznego interesu. Już w tej chwili widać, że „pierwsza liga" firm komputerowych na rynku polskim w strategię swojego rozwoju włożyła dokładne rozpoznanie potrzeb klienta. Gdy się spojrzy na formy prezentacji firm komputerowych na naszych spotkaniach, na charakter dyskusji i materiałów propagandowych czy edukacyjnych – można wyraźnie dostrzec korzyści, jakie odnosi z tego użytkownik. Jedną z nich jest uzgodnienie między największymi dostawcami (HP, IBM, ICL, DEC) standardu systemu otwartego. Stowarzyszenie wydało ten standard w postaci publikacji. Złożoność technologii informatycznej sprawia, iż potrzebny jest dialog użytkownika z dostawcą, potrzebne jest też istnienie silnego lobby użytkownika. Stowarzyszenie chce pilnować interesów użytkowników nie przez oskarżanie firm, że są „be", ale raczej przez takie kształtowanie warunków organizacyjno-prawnych, by z góry eliminować nieuczciwych dostawców, np. przez określone procedury projektowania systemów.

– Na przełomie sierpnia i września środowisko informatyczne poruszone zostało artykułami zamieszczonymi w „Życiu Warszawy", sugerującymi aferę korupcyjną związaną z budową systemu informatycznego w GUC. Czy nie obawia się Pan, że takich spraw może być coraz więcej?

– Jeszcze cztery lata temu, duży przetarg zaczynał się od 2 mld, dziś mamy przetargi na kilkadziesiąt miliardów, a w niektórych kontraktach i powyżej 200 mld. Wcale mnie nie dziwi, że przy ogromnej niedoskonałości naszych różnych procedur przetargowych i przy takiej skali wydatków, różni ludzie mogą mieć wątpliwości czy wszystko było O.K. Czekając na ustawę o zamówieniach publicznych, Biuro do Spraw Informatyki URM sformalizowało procedury przetargów, wręcz przetłumaczyliśmy niektóre fragmenty procedur z krajów EWG i upowszechniliśmy je jako zalecenia. Mam nadzieję, że po wejściu w życie ustawy (1 stycznia 1995 r.), nie będzie podstaw do szukania afer.

– W jaki sposób SRSO realizuje swoją rolę edukacyjną?

— Temu celowi służą przede wszystkim imprezy organizowane przez Stowarzyszenie. Jest ich już bardzo dużo, kilkanaście w roku. Gromadzą one zawsze więcej niż 200–300 osób. Uczestniczymy w targach Infosystem, Komputer Expo i Infoman, gdzie Stowarzyszenie przejęło patronat nad nagrodą im. Stanisława Staszica, czyli formą promocji polskich domów software'owych. Wydaliśmy w dużym nakładzie „Charakterystykę Systemów Otwartych". Na I Kongres Informatyki Polskiej przygotowujemy analizę szans i zagrożeń polskiej informatyki, właśnie w interesie użytkowników. Znakomicie uzupełnia ona analizy przygotowywane przez inne organizacje, dotyczące dostawców i twórców informatyki. Analizy te w sposób jasny określają interesy poszczególnych grup, które często są ze sobą sprzeczne. To najlepszy dowód na to, że gdyby nie było Stowarzyszenia – należałoby je wymyślić.

– W jaki sposób Stowarzyszenie chce wspomagać poszczególne grupy użytkowników?

– Sądzę iż zaczną powstawać sekcje branżowe. Są ludzie zainteresowani oceną sprzętu komputerowego, telekomunikacyjnego, drukarek itp. Ogromne emocje budzi oprogramowanie służące zarządzaniu, a zwłaszcza programy f-k. W nowym typie umów chcemy wprowadzić klauzulę, że muszą być zdeponowane ich kody źródłowe, by użytkownik mógł dostosowywać program do nowych przepisów. To jedna z podstawowych zasad systemu otwartego.

– Między interesami członków poszczególnych stowarzyszeń istnieją wyraźne konflikty interesów, chociażby sprawa walki z pirackim oprogramowaniem, które można dziś spotkać w każdej nieomal instytucji używającej komputerów. Ich użytkownicy głoszą nieraz pogląd, że chronią w ten sposób swoją firmę przed nadmiernymi wydatkami. Jakie są możliwości uzgadniania sposobów ich rozwiązywania dla wspólnego dobra?

– Samochód służbowy zwiększa efektywność działania urzędnika, bo pozwala mu przenosić się z miejsca na miejsce. Czy można więc powiedzieć, że zakup samochodu obciąża podatnika i wobec tego lepiej byłoby jeździć kradzionym? Dla mnie oprogramowanie jest takim samym środkiem technicznym jak sprzęt. Ale to prawda, są jeszcze ludzie, którzy uważają, że kradzież dotyczy tylko rzeczy materialnej. W tym przypadku i w wielu innych nie ma konfliktu interesów, lecz mamy do czynienia z zafałszowaną, jeszcze niewykształconą świadomością społeczną. Spotkania między stowarzyszeniami będą służyły wyartykułowaniu konfliktów interesów i szukaniu platformy porozumienia.

– Czego Stowarzyszenie oczekuje po Kongresie i na jakie problemy będzie kłaść nacisk?

– Spodziewamy się jasnej, wyraźnej konfrontacji z dwoma grupami interesów: dostawcami dóbr informatycznych i z tymi, którzy reprezentują świat nauki, dydaktyki i techniki. Chciałbym, by doszło do krystalizacji poglądów na temat strategii zastosowania informatyki w Polsce. Musi się ona oprzeć na tych trzech grupach, a więc trzeba sprecyzować poglądy na temat wspólnoty interesów i rozbieżności. Powinniśmy odpowiedzieć też na jedno z podstawowych pytań: czy informatyka może być częścią reformy ekonomii polskiej jako całości? Z tym wiąże się następne pytanie: czy nasza gospodarka jest w stanie wytwarzać dobra informatyczne, czy jedynie je nabywać? Wchodzą do nas bardzo zaawansowane technologie zachodnie i nie jest obojętne czy transfer technologii odbywa się przy udziale polskich podmiotów gospodarczych czy też bez nich. Użytkownik ma prawo spojrzeć na to z własnego punktu widzenia. Domagamy się bowiem wysokiej jakości produktów po umiarkowanych cenach. Dzisiaj widzę bardzo wyraźnie szanse na to, by coraz większa część wartości dodanej w informatyce powstawała w Polsce. Potrzebne są decyzje rządowe, ale nie w postaci stwarzania sztucznych barier celnych, jak się to dzieje w branży motoryzacyjnej. Kongres, mam nadzieję pokaże, że nasza gospodarka jest w stanie uczestniczyć w wytwarzaniu dobrych produktów informatycznych na własne potrzeby i jeszcze je eksportować. Proszę zauważyć, że podstawowe wielkie firmy polskie zaczynają wdrażać normy ISO 9000. Bez świadomego nacisku na jakość ze strony klientów nie byłoby takiego zainteresowania.

– Dziękuję Panu za rozmowę.

O standaryzacji w zakresie technik kryptograficznych

Janusz Stokłosa
Politechnika Poznańska

Narodowe Biuro Standardów USA ogłosiło w 1973 r. konkurs na opracowanie algorytmu kryptograficznego do ochrony danych z przeznaczeniem do stosowania poza sferą dyplomacji i wojskowości. Przedłożony przez grupę badaczy z firmy IBM algorytm uzyskał w 1977 r. status standardu państwowego FIPS (*Federal Information Processing Standard*) i dzisiaj jest znany jako DES (*Data Encryption Standard*) [3].

Od tego czasu standaryzacja w zakresie technik kryptograficznych rozwija się intensywnie. Celem tej standaryzacji jest przede wszystkim zaoferowanie bezpiecznych rozwiązań w zakresie:

- algorytmów i protokołów kryptograficznych[1],
- trybów stosowania algorytmów i protokołów,
- zastosowań.

Jeśli chodzi o algorytmy i protokoły podlegające standaryzacji, to istotne jest, że są one powszechnie znane, a ich siła zależy wyłącznie od klucza kryptograficznego. Sprzyjającą okolicznością jest możliwość wszechstronnego ich przebadania przed wprowadzeniem do użytku. Mówiąc o trybach stosowania algorytmów i protokołów mamy na uwadze specyfikacje takiego ich stosowania, które zapewnia zarówno tajność, jak i autentyczność danych. Istotny jest tu także problem zarządzania kluczami kryptograficznymi. Zastosowania kryptografii obejmują ochronę danych w zakresie ich przesyłania i gromadzenia. W pracach organizacji standaryzacyjnych odnoszą się przede wszystkim do dziedzin telekomunikacji oraz bankowości.

Organizacje standaryzujące

Normalizacją w zakresie kryptografii zajmują się organizacje międzynarodowe (o zasięgu ogólnoświatowym), regionalne, krajowe i inne. Wśród organizacji międzynarodowych główną rolę odgrywają ISO (*International Organization for Standarization*), IEC (*International Electrotechnical Commission*) oraz CCITT (*Comité Consultatif International Télégraphique et Téléphonique*). Należy tu jeszcze wspomnieć o Radzie Gospodarczo-Społecznej ONZ, pod auspicjami której są opracowywane normy dla elektronicznej wymiany danych (EDI). Do tego celu powołano grupę roboczą UN/EDIFACT *Security Joint Working Group*.

ISO/IEC

W ISO zadania normalizacyjne są wykonywane w Komitetach Technicznych (*Technical Committee* – TC), które dzielą się na podkomitety (*Subcommittee* – SC), a te z kolei na grupy robocze (*Working Group* – WG); niekiedy powołuje się specjalne grupy robocze (*Special Working Group* – SWG). Zanim dojdzie do opublikowania normy mającej status standardu międzynarodowego (*International Standard* – IS), powstają cztery dokumenty pośrednie. Najpierw jest to propozycja (*New work item Proposal* – NP), następnie szkic roboczy (*Working Draft* – WD), później dokument oznaczany jako CD (*Committee Draft*), wreszcie szkic standardu międzynarodowego (*Draft International Standard* – DIS). Po przyjęciu DIS dokument ten zostaje opublikowany jako standard międzynarodowy. W każdym etapie tego procesu do narodowych komitetów normalizacyjnych (w Polsce jest to obecnie Polski Komitet Normalizacyjny) przesyłane są dokumenty w celu ich zaopiniowania. Głosuje się nad przyjęciem NP, CD i DIS. Procedura ta wymaga czasu i w najlepszym przypadku od chwili zgłoszenia propozycji do momentu opublikowania normy upływają trzy lata.

W 1987 r. ISO i IEC utworzyły Połączony Komitet Techniczny Nr 1 (*Joint Technical Committee No.1* – JTC 1) o nazwie „Technologia informacyjna" (*Information Technology* – IT), w ramach którego współpracują one przy opracowywaniu norm obejmujących zakres jego kompetencji. Dzisiaj w składzie JTC 1 pracuje równorzędnie 19 podkomitetów (SC), jedna grupa robocza (WG) oraz trzy specjalne grupy robocze. Są wśród nich (pozostawiam nazwy angielskie):

- SC1 – *Vocabulary*,
- SC6 – *Telecommunications & Information Exchange Between Systems*,
- SC7 – *Software Engineering*,
- SC17 – *Identification Cards' Related Devices*,
- SC18 – *Document Processing and Related Communication*,
- *SC21* – *Open System Interconnection – Data Management and Open Distributed Processing*,
- SC27 – IT *Security Techniques*,
- WG3 – *Open Electronic Data Interchange*.

W ISO istnieje także komitet TC68, którego terenem działalności jest bankowość („*Banking and Related Financial Services*"), w tym także ochrona danych. Zajmuje się tymi sprawami również w podkomitecie SC2 grupa robocza WG2 („*Security for Wholesale Banking*") oraz WG7 („*Security Architecture of Banking Systems using the Integrated Circuit Card*").

CCITT

CCITT wchodzi w skład Międzynarodowej Unii Telekomunikacyjnej (ITU), a tym samym plasuje się w strukturze ONZ. Terenem działalności Komitetu jest telefonia i systemy przesyłania danych. Powstają tam normy techniczne, nazywane w oficjalnej nomenklaturze zaleceniami (rekomendacjami). Opracowano rekomendacje z serii V (np. V.24, V.32), X (np. X.25, X.32, X.400), I, Q, T i inne. Prace związane z bezpieczeństwem danych są prowadzone w grupie studyjnej (*Study Group*) SG VII („*Data Networks*").

Bardzo aktywne są regionalne organizacje normalizacyjne w USA oraz w głównych krajach europejskich, i chociaż nie mają one

[1] W rzeczy samej każdy protokół jest algorytmem. Jednakże protokół oznacza tu procedurę stosowaną do wymiany informacji w systemie rozproszonym, natomiast o algorytmie mówimy wtedy, gdy mamy na uwadze przetwarzanie lokalne.

Prof. JANUSZ STOKŁOSA pracuje w Politechnice Poznańskiej. Jego zainteresowania naukowe obejmują szczególnie algorytmy kryptograficzne i kryptograficzne metody ochrony w sieciach komputerowych. Jest ekspertem w Zespole Informatyki i Telekomunikacji (podkomitet „Zabezpieczenia systemów i ochrona danych") Polskiego Komitetu Normalizacyjnego.

charakteru międzynarodowego (w przyjętym tego słowa znaczeniu), to ich oddziaływanie, szczególnie tych w USA, jest ogólnoświatowe.

ANSI

Główne komitety (*Accredited Standards Committee* – ASC) działające w zakresie technologii informatycznej w Narodowym Amerykańskim Instytucie Standaryzacyjnym (*American National Standards Institute* – ANSI) to: JTC 1 TAG – techniczna grupa doradcza (*Technical Advisory Group* – TAG), współpracująca z ISO/IEC JTC 1, ASC T1 – odpowiedzialna za współpracę pomiędzy Departamentem Stanu i CCITT, a także ASC X12 – opracowująca standardy w zakresie elektronicznej wymiany danych (EDI).

IEEE

Instytut Inżynierów Elektryków i Elektroników (*Institute of Electrical and Electronics Engineers*) jest prawdopodobnie największą organizacją zawodową na świecie. Pracuje się tam nad normami w zakresie elektroniki i informatyki, w tym dotyczącymi lokalnych sieci komputerowych (trzeba tu wymienić standardy IEEE 802.x) oraz interfejsu POSIX. Co do sieci lokalnych, to opracowano normę IEEE 802.10, znaną pod akronimem SILS (*Standard for Interperable* LAN *Security*) [9, 10], obejmującą m.in. takie zagadnienia, jak bezpieczna wymiana danych w warstwie łącza danych i zarządzanie kluczami kryptograficznymi.

NIST

Krajowy Instytut Standardów i Technologii (*National Institute of Standards and Technology*), działający poprzednio jako Krajowe Biuro Standardów (*National Bureau of Standards* – NBS), wydaje w USA normy o nazwie *Federal Information Processing Standard* (FIPS). Znane są tu dokumenty dotyczące wspomnianego na wstępie algorytmu DES [3] (przyjęty przez ISO w 1987 r. jako DEA – *Data Encryption Algorithm* [12]) i sposobów jego stosowania [4], uwierzytelniania [6], podpisów cyfrowych [8], funkcji haszowej [7], czy używania haseł [5].

ETSI

Europejski Instytut Standardów Telekomunikacyjnych (*European Telecommunications Standards Institut*), utworzony przez Europejską Komisję Ekonomiczną (*European Economic Commision*), zajmuje się standaryzacją kart inteligentnych (ang. *smart cards*), bezpieczeństwem systemów obsługi wiadomości (*Message Hendling System* – MHS; rekomendacja X.400), zarządzaniem sieciami, usługami ruchomymi.

ECMA

Do Europejskiego Zrzeszenia Producentów Komputerów (*European Computer Manufacturers Association*) należą takie firmy, jak IBM, DEC, Honeywell Bull, Siemens, Univac, Burroughs i in. Prowadzi się tu prace normalizacyjne w zakresie bezpieczeństwa systemów baz danych, bezpieczeństwa w dolnych warstwach modelu OSI, bezpieczeństwa systemów otwartych. W ramach projektu SASAME (*Secure European System for Applications in a Multivendor Environment*) rozwijana jest architektura implementująca model ECMA [28].

Należy jeszcze wskazać inne organizacje, przede wszystkim rządowe, które opublikowały dokumenty standaryzujące kryteria ewaluacji systemów komputerowych w zakresie bezpieczeństwa. Do tej grupy należy Departament Obrony Stanów Zjednoczonych, który wespół z Narodowym Centrum Bezpieczeństwa Komputerowego (*National Computer Security Center* – NCSC) wydał następujące trzy dokumenty:

- *Trusted Computer System Evaluation Cryteria* (TCSEC) [2],
- *Trusted Network Interpretation* (TNI) [26],
- *Trusted Database Interpretation* (TDI); treść tego dokumentu jest tajna.

W Europie Holandia, Francja, Niemcy i Wielka Brytania opracowują własne kryteria bezpieczeństwa: *Information Technology Security Evaluation Criteria* [25].

Bezpieczeństwo w systemach otwartych

Komitet ISO/IEC JTC 1 przez swoje podkomitety SC6, SC21 i SC27, uczestniczy w standaryzacji obejmującej architekturę zabezpieczeń, struktur, protokoły i usługi, techniki kryptograficzne, modele i zastosowania.

Standard ISO 7498 z 1983 r. definiuje Model Odniesienia Systemów Otwartych OSI. Jego druga część, opublikowana cztery lata później, normalizuje architekturę zabezpieczeń [16] (istnieje polski odpowiednik tej normy [27]). Zabezpieczenia kryptograficzne można stosować we wszystkich warstwach modelu OSI poza warstwą sesji. Wymagania w zakresie bezpieczeństwa w warstwie fizycznej zawarto w normie ISO 9160 [17]. Co do dokumentów odnoszących się do innych warstw, to w fazie standaryzacji w JTC 1 SC6, znajdują się normy obejmujące protokół bezpieczeństwa w warstwie transportowej [23] i protokół bezpieczeństwa w warstwie sieciowej [24].

Ważnymi standardami odnoszącymi się do bezpieczeństwa w zastosowaniach rozproszonych są X.400 i X.509. W wersji z 1988 r. rekomendacji X.400 omówiono następujące zabezpieczenia: uwierzytelnianie, integralność, poufność i nie zaprzeczanie (ang. *non-repudation*). Można je stosować zarówno w poczcie elektronicznej, jak i w elektronicznej wymianie dokumentów (EDI). Podobne rozwiązania zostały zaadaptowane przez ISO/IEC JTC 1/SC18 i są znane jako MOTIS. W rekomendacji X.400 zalecenia dotyczące mechanizmów uwierzytelniania z użyciem haseł i z użyciem technik szyfrowania asymetrycznego. W ISO przyjęto je w postaci standardu ISO/IEC 9594-8 [18]. Należy zauważyć, że funkcja haszowa zaproponowana w aneksie D tej normy nie jest bezpieczna (por. dokument ISO/IEC JTC 1/SC20/WG2/N160). Mechanizmy uwierzytelniania opisane w X.509 krytykowano kilkakrotnie, m.in. w [1].

Co do zarządzania systemami, to w opracowaniu jest norma określająca zasady alarmowania przy zagrożeniu bezpieczeństwa (DIS 10164-6).

ISO/IEC TJC 1/SC27

Jak już wyżej wspomniano w 1987 r. ISO oraz IEC podjęły wspólne wysiłki w zakresie normalizacji ukierunkowanej na technologię informacyjną. Po upływie dwóch lat, w 1989 r. w ramach JTC 1 zastąpiono podkomitet SC20 („*Data Cryptographic Techniques*") przez SC27 („*Security Techniques*"). Podkomitet SC27 uzyskał więc szersze pole działania. W jego ramach powołano trzy grupy robocze:

- WG1 – Wymagania, Usługi w Zakresie Zabezpieczeń i Wskazania,
- WG2 – Techniki i Mechanizmy Zabezpieczeń,
- WG3 – Kryteria Oceny Zabezpieczeń.

Zadaniem WG1 jest opracowywanie norm odnoszących się do usług w zakresie zabezpieczeń, np. uwierzytelniania, kontroli dostępu, integralności, monitorowania, z użyciem technik i mechanizmów opracowanych w WG2, a ponadto identyfikacja wymagań oraz praca nad dokumentami wspomagającymi (glosariusze, wskazania w zakresie zabezpieczeń, analiza ryzyka).

WG2 jest centrum dokonującym ekspertyz w zakresie standaryzacji technik i mechanizmów zabezpieczeń. Mogą one mieć naturę kryptograficzną i pozakryptograficzną (np. w odniesieniu do kontroli dostępu i haseł).

Standardy związane z kryteriami oceny oraz certyfikacji systemów, komponentów i produktów są opracowywane w ramach WG3. W zakres prac wchodzą dokumenty dotyczące sieci komputerowych, systemów rozproszonych, aplikacji itp. Bierze się tu pod uwagę trzy aspekty: kryteria oceny, metodologię stosowania tych kryteriów, procedury administracyjne dokonywania oceny, certyfikacji i akredytacji.

Obecnie główne zadania poszczególnych grup roboczych są związane z:

- w **WG1:**
  - mechanizmami uwierzytelniania (IS 9798-1 [20], rewizja w 1995 r.),
  - zarządzaniem kluczami (norma kilkuczęściowa, aktualnie CD 11770, IS przewidywany w 1995 r.),
  - procedurami rejestracji algorytmów kryptograficznych (rewizja przyjętej w 1991 r. normy IS 9979 [21] – 1995 r.),
  - wskazaniami w zakresie zarządzania zabezpieczeniami (trójczęściowa norma 13335, IS w 1995 r.),
  - obiektami zabezpieczeń (IS w 1995 r.)
- w **WG2:**
  - trybami działania algorytmów szyfrowania blokowego (rewizja IS 8372 z 1987 r. [11] w 1996 r. i rewizja IS 10116 [22] w 1994 r.),
  - mechanizmami uwierzytelniania (wieloczęściowy standard 9798),
  - mechanizmami integralności, wykorzystującymi techniki wiedzy zerowej,

– kwestiami odnoszącymi się do zaprzeczeń (na 1995 r. jest planowane opracowanie 3-częściowej normy obejmującej · model ogólny, zastosowanie algorytmów szyfrowania symetrycznego i asymetrycznego),
– podpisami cyfrowymi (m.in. rewizja IS 9796 [19] w 1995 r.),
– funkcjami haszowymi (norma 4-częściowa, aktualnie WD 10118),

■ w **WG3:**
– kryteriami oceny zabezpieczeń (3-częściowy IS w 1995 r.),
– terminologią w zakresie zabezpieczeń.

W Polskim Komitecie Normalizacyjnym zawiązano na przełomie 1993 i 1994 r. odpowiednik JTC 1 w postaci VI Tematycznej Grupy Roboczej „Zabezpieczenia systemów i ochrona danych", dzieląc ją na trzy podgrupy odpowiadające WG1, WG2 i WG3. W ten sposób Polska włączyła się w tok prac na forum ISO/IEC.

ISO/TC68

Prace prowadzone w Komitecie Technicznym TC68 obejmują bankowość i związane z nią usługi finansowe [29]. Opracowano standardy w zakresie uwierzytelniania wiadomości i zarządzania kluczami. Trzeba tu wymienić IS 9807 [15], zawierający wymagania w zakresie uwierzytelniania wiadomości oraz dwuczęściową normę IS 8731 [12, 13], definiującą algorytmy uwierzytelniania wiadomości. Zarządzanie kluczami kryptograficznymi opisano w IS 8732 [14], ale problematykę tę obejmują również normy 11166 i 11568. Ponadto standaryzuje się zarządzanie osobistymi numerami identyfikacyjnymi (PIN – *Personal Identification Number*) co uczyniono w IS 9564.

* * *

Przedstawione tu kwestie z zakresu standaryzacji technik kryptograficznych nie wyczerpują tej tematyki. Są to zagadnienia bardzo ważkie. Mają swoje odbicie w bieżącej działalności banków, firm handlowych i organizacji rządowych. Dzisiaj niektóre dane mają szczególnie dużą wartość. Nie można sobie pozwolić na ich przywłaszczenie (kradzież). Dobrze, że w coraz większym zakresie uświadamiają to sobie decydenci także w naszym kraju.

LITERATURA

[1 Burrows M., Abadi M., Needham R.: A logic of authentication. Report No. 39. Digital Systems Research Center, Palo Alto, CA, 1989
[2] DoD 5200.28-STD „Trusted Computer Systems Evaluation Cryteria". Department of Defence. 1985
[3] FIPS PUB 46 „Data Encryption Standard". National Bureau of Standards, US Department of Commerce, Washington DC, 1977
[4] FIPS PUB 81 „DES Modes of Operation". National Bureau of Standards, US Department of Commerce, Washington DC, 1980
[5] FIPS PUB 112 „Password Usage". National Bureau of Standards, US Department of Commerce, Washington DC, 1985
[6] FIPS PUB 113 „Computer Data Authentication". National Bureau of Standards, US Department of Commerce, Washington DC, 1985
[7] FIPS PUB 180 „Secure Hash Standard". National Bureau of Standards, US Department of Commerce, Washington DC, 1993
[8] FIPS PUB xxx „Digital Signature Standard". National Bureau of Standards, US Department of Commerce, Washington DC, 1991
[9] IEEE 802.10 Standard, Part B – Secure Data Exchange. November 9, 1991
[10] IEEE 802.10 Standard, Part C – Key Management. April 10, 1992
[11] ISO 8372 „Information processing – Data cryptographic techniques – Modes of operation for a 64-bit block cipher algorithms". ISO, 1987
[12] ISO 8731-1 „Banking – approved algorithms for message authentication – Part 1: DEA". ISO, 1987
[13] ISO 8731-2 „Banking – approved algorithms for message authentication – Part 2: Message Authentication Algorithm (MAA)". ISO 1987
[14] ISO 8732 „Banking – Key management (wholesale)". ISO, 1988
[15] ISO 9807 „Banking and related financial services – Requirements for message authentication". ISO/IEC, 1991
[16] ISO/IEC 7498-2 „Information processing systems – Open System Interconnection – Basic Reference model Part 2: Security Architecture. ISO, 1989
[17] ISO/IEC 9160 „Information processing – Data encipherment – Physical leyer interoperability requirements". ISO/IEC, 1988
[18] ISO/IEC 9594-8 „Information technology – Open system interconnection – The directory – Part 8: Authentication framework". ISO/IEC, 1990
[19] ISO/IEC 9796 „Information technology – Security techniques – Digital signature scheme giving message recovery". ISO/IEC, 1991
[20] ISO/IEC 9798-1 „Entity authentication mechanisms, Part 1: General Model". ISO/IEC, 1991
[21] ISO/IEC 9979 „Data cryptographic techniques – Procedures for the registration of cryptographic algorithm". ISO/IEC, 1991
[22] ISO/IEC 10116 „Information technology – Modes of operation for n-bit block cipher algorithm". ISO/IEC, 1991

dokończenie na s. 28

Dokładna nazwa i adres firmy:

NIP .....

Adres do korespondencji:

..... dnia .....1994r. L.dz. ...../94

DEKLARACJA UCZESTNICTWA
w I KONGRESIE INFORMATYKI POLSKIEJ

Deklaruję udział w Kongresie:

Pan(i): ....
imię i nazwisko tel. fax

w grupie środowiskowej:

UZ - użytkownicy * ES - edukacja i szkolenia * PD - producenci i dostawcy *

oraz wybieram udział w sesji roboczej następującej grupy tematycznej:

A * - rządowe przedsięwzięcia informatyczne
B * - warunki rozwoju polskiego oprogramowania
C * - świadomość społeczna w odbiorze informatyki
D * - wypełnianie potrzeb kadrowych w informatyce
E * - powiązania pomiędzy rozwojem telekomunikacji i informatyki
F * - nowe technologie informatyczne

Proszę o dokonanie rezerwacji (1/2 i 2/3.XII.1994r. - 2 doby) w wybranym (w.g. oferty organizatorów) hotelu:

hotel POZNAŃ * hotel MERKURY * hotel PAN *

Zamawiam miejsce w pokoju: jednoosobowym * dwuosobowym *

Zobowiązuję się do:
- wpłacenia należności z tytułu kosztów uczestnictwa w wysokości 2,5 mln zł - w terminie 7 dni od daty otrzymania faktury
- uregulowania należności z tytułu rezerwacji hotelowej w kwocie wynikającej ze stawki wybranego hotelu - w terminie 7 dni od daty otrzymania faktury.

*niepotrzebne skreślić

podpis osoby upoważnionej

BIURO ORGANIZACYJNE 00-503 Warszawa 15, skr. p. 44
ul. Żurawia 4a
tel. 693-59-46, 621-76-26
fax 693-59-49

Podprogramy w języku symulacji systemów dynamicznych AMIL

Marek Stabrowski
Instytut Elektroniki Teoretycznej
i Miernictwa Elektrycznego
Politechnika Warszawska

Język symulacji układów dynamicznych AMIL, jego kompilator i otoczenie zintegrowane kompilatora [5, 6] są użytecznymi i wygodnymi narzędziami analizy zjawisk, obiektów i konstrukcji opisywanych za pomocą układów równań różniczkowych i algebraicznych. Istotną zaletą systemu symulacyjnego AMIL jest jego zintegrowane otoczenie, prowadzące użytkownika do edycji postaci źródłowej programu, przez kompilację i realizację eksperymentu symulacyjnego, aż do graficznego przetwarzania wyników symulacji. Jedną z ostatnich modyfikacji systemu było wprowadzenie przez autora tego artykułu rozbudowanej interakcji w graficznym przetwarzaniu wyników eksperymentu [6].

Jednakże nawet po wprowadzeniu tych zmian, system symulacyjny AMIL ustępował innym współczesnym językom i systemom symulacji układów dynamicznych [1, 3], co najmniej z dwóch powodów:

- brakowało w nim mechanizmu definiowania i używania podprogramów, nazywanych w systemach symulacyjnych submodelami,
- reprezentacja podstawowych operacji i danych nie zawsze była w pełni logiczna, a właściwie nie była zgodna ze współczesnymi tendencjami [1, 3] zapisu równań różniczkowych w językach symulacyjnych.

Celem tego artykułu jest przedstawienie implementacji mechanizmu definiowania, kompilacji, konsolidacji i wykonywania podprogramów w systemie symulacyjnym AMIL. Koncepcje tej implementacji nie są oparte na jakichś nowych ideach. Należy jednak podkreślić, że zaprezentowano w nim całość praktycznie sprawdzonego rozwiązania, a tym samym jego użyteczność w opracowaniu nowego oprogramowania tego typu wydaje się nie ulegać wątpliwości. Odwołując się, niezbyt elegancko do terminologii angielskojęzycznej, można zakwalifikować ten artykuł do grupy artykułów przedstawiających tzw. *case story*. Należy dodać, że przed opisanymi zmianami w języku, kompilatorze i środowisku zintegrowanym, system AMIL był zrealizowany jako produkt hybrydowy, napisany przede wszystkim w Fortranie, a częściowo w języku C. Przedstawionym tu zmianom i rozszerzeniom języka AMIL towarzyszyła konwersja całości na język C, obejmująca również część numeryczną.

Definiowanie i używanie podprogramów-submodeli w języku AMIL

Podprogramy-submodele w językach symulacji układów dynamicznych pełnią podobne funkcje, jak podprogramy w uniwersalnych językach algorytmicznych (Fortran, Pascal, C), są też podobnie definiowane i używane. Występują tu jednak także istotne różnice, zwłaszcza w mniej interesującej użytkownika fazie kompilacji i konsolidacji.

Zasadniczym zadaniem podprogramów-submodeli jest dekompozycja dużych monolitycznych programów (modeli), gdyż oczywiście mniejsze moduły programowe łatwiej można kontrolować, modyfikować i używać. W języku AMIL przyjęto zasady definiowania i używania podprogramów-submodeli podobne (choć nie identyczne) do zasad obowiązujących w dwóch współczesnych językach symulacyjnych – ESL [1] oraz COSMOS [3]. Zasady te są zarazem bardzo zbliżone do konwencji przyjętych w języku C.

Strukturę programu w języku AMIL, zawierającego dwa podprogramy, przedstawiono w poniższym przykładzie. Ze zrozumiałych względów nie jest istotny układ [2] i problem modelowany za pomocą tego programu, chodzi tu raczej o przykład składni tych nowych konstrukcji języka. Program ten składa się z modułu głównego, zawierającego część opisującą model i część opisującą eksperyment, oraz z dwóch podprogramów. Poszczególne podprogramy są zawarte wewnątrz ograniczników BSM (ang. *Begin SubModel*) oraz ESM (ang.

```
* prosty uklad z dwiema wewnętrznymi i jedną zewnętrzną
* pętlą sprzężenia zwrotnego
* eksperyment
        CON (k1=1.0, k2=1.0, k3=10.0, k4=10.0, k5=10.0)
        CON (dt=0.02, tstop=1.00)
        CON (v0=0.0, out0=0.0)
        CON (e1=0.2)
        FIN (t, tstop)
        STO (t, in, outp)
        PRI (t, in, outp)
* moduł modelu głównego
        in = SIN (2.0*t)
        e1=imp(e1, in-fb3)
        LOOP1(e1, v0, v)
        outp = LOOP2 (v, out0)
        fb3 = k5*outp
* pierwszy submodel - integrator z ujemnym sprzężeniem zwrotnym
        BSM
        LOOP1 (input, in0, ou)
*       wejścia submodelu
        INP (input, in0)
        ou = INT(edif, in0)
        efdb = k3*ou
        edif = input - efdb
        ESM
*       koniec definicji pierwszego submodelu
* drugi submodel - integrator z ujemnym sprzężeniem zwrotnym
        BSM
        LOOP2 (input, in0)
*       wejścia submodelu
        INP (input, in0)
        ou = INT(edif, in0)
        efdb = k4*ou
        edif = input - efdb
        RET(ou)
        ESM
*       koniec definicji drugiego submodelu
        END
```

End SubModel). AMIL jest językiem paralelnym, tzn. kolejność instrukcji i deklaracji w danym module programowym jest dowolna, ale moduły powinny tworzyć uporządkowaną sekwencję, tj. pierwszym modułem powinien być program (model, system), poprzedzający wszystkie podprogramy-submodele.

Podany na wydruku przykład pozwala zauważyć, że podprogram w języku AMIL jest w zasadzie podprogramem typu funkcji (w tym przykładzie – LOOP2), zwracającym na zewnątrz (instrukcja *RET* – ang. *RETurn*) pojedynczą wartość zmiennej. Jeśli w podprogramie-submodelu nie użyto instrukcji *RET*, to jest on podprogramem typu procedury, przekazującym na zewnątrz określone wartości przez argumenty wywołania. Tego rodzaju podprogram-submodel (w przykładzie – LOOP1) nie może oczywiście wystąpić po prawej stronie instrukcji podstawienia, czy też jako element wyrażenia arytmetycznego.

W podanym przykładzie programu widać, że w języku AMIL nie wprowadzono prototypów podprogramów (zresztą i zmienne nie są deklarowane w sposób jawny). Jako swego rodzaju prototyp jest traktowana definicja podprogramu, tzn. pierwszy różny od komentarza wiersz występujący po instrukcji BSM-ograniczniku początkowym podprogramu. Zgodnie z ogólną koncepcją języka AMIL, wszystkie zmienne są w nim typu rzeczywistego (zmiennopozycyjne) i podprogramy zwracają na zewnątrz (jeśli zawierają instrukcję *RET*) również wartości typu rzeczywistego. Z uwag dotyczących podprogramu-submodelu LOOP1 wynika, że w języku AMIL przyjęto zasadę przekazywania argumentów do wnętrza podprogramów nie jako wartości, a jako adresów. Aby wyeliminować dość znany (np. klasyczny przykład w weryfikatorze PROFORT), a pojawiający się przy takiej konwencji komunikacji z podprogramami problem ewentualnego modyfikowania literałów (stałych liczbowych) używanych jako argumenty aktualne, wprowadzono ograniczenie, że argumentami mogą być tylko zmienne, a nie literały czy wyrażenia arytmetyczne.

Zmienne zdefiniowane (przez użycie lub wystąpienie w deklaracji *CON* – ang. *CONstants*) w programie głównym są widziane i dostępne we wszystkich podprogramach, a więc mają charakter globalny. Nie przewidziano możliwości ich „przysłonięcia" przez zmienne lokalne danego modułu o identycznych nazwach. Zaleca się, podobnie jak w wielu językach programowania, ograniczenie używania zmiennych globalnych w podprogramach-submodelach do niezbędnego minimum. Deklaracja *INP* (ang. *INPuts*) definiuje określone argumenty podprogramu jako wejściowe, tzn. takie, które powinny być obliczone przed użyciem danego podprogramu. Sposób wykorzystania informacji zawartej w deklaracji INP zostanie szczegółowo omówiony przy okazji analizy problemów sortowania kodu pośredniego.

Kompilacja podprogramów-submodeli oraz ich wywołań

Przed omówieniem zagadnień kompilacji, konsolidacji i sortowania warto wspomnieć krótko o systemie jednolitych identyfikatorów poszczególnych obiektów leksylalnych, składających się na program. W zasadzie każdemu obiektowi (zmienna, literał, funkcja standardowa itp.) przypisano dwa identyfikatory – identyfikator klasy oraz identyfikator obiektu w obrębie klasy, czyli po prostu numer porządkowy. Wyróżniane przez kompilator języka AMIL klasy obiektów są następujące:

0 – zmienne i stałe (*CON*) globalne i lokalne;
1 – literały (stałe liczbowe);
2 – funkcje standardowe języka;
3 – łańcuchy znakowe, występujące jako argumenty funkcji definiujących tytuły tabel liczbowych lub wykresów (*HDR, TTX* itd.);
4 – argumenty formalne podprogramów-submodeli;
5 – podprogramy-submodele – przed konsolidacją podklasa 5–9, a po konsolidacji – podklasa 0–4;
6 – wskaźniki ramek z listami argumentów aktualnych.

Jak widać, dla potrzeb definiowania i przetwarzania podprogramów wprowadzono trzy nowe klasy obiektów.

Kompilacja podprogramu, po napotkaniu ogranicznika *BSM* sygnalizującego jego początek, polega na generowaniu kodu pośredniego [5, 6], tak jak w przypadku modułu głównego, a także na wypełnieniu informacjami struktury *ALL_SUBROUTS* (rys. 1), przypisanej do danego podprogramu-submodelu. Pierwszym elementem tej struktury jest nazwa podprogramu – łańcuch znakowy, natomiast pozostałymi elementami – liczby całkowite. Adres jest po prostu numerem wiersza kodu pośredniego, w którym rozpoczyna się kod podprogramu, a liczba argumentów – liczbą argumentów formalnych zidentyfikowanych w definicji. Nazwy argumentów formalnych są zapisywane w odrębnej tablicy *NAMARGS*. Element „pierwszy argument" ze struktury *ALL_SUBROUTS* jest indeksem tablicy *NAMARGS* i ma oczywiście dodany identyfikator klasy obiektów równy 4. Kolejny element – „argument zwrotny" (rys. 1) – jest indeksem zmiennej lokalnej lub globalnej o wartości przechowywanej w tablicy VARCON [] i identyfikatorze klasy równym 0. Elementowi temu można nadać właściwą wartość dopiero po napotkaniu w podprogramie instrukcji *RET*. Gdy takiej instrukcji brak, to element ten ma wartość 0, sygnalizując, że podprogram jest typu procedury, bez argumentu zwrotnego. Dwa kolejne elementy struktury *ALL_SUBROUTS* są identyfikatorami pierwszej i ostatniej zmiennej lokalnej danego podprogramu, przy czym wartość drugiego z tych elementów jest znana po napotkaniu końcowego ogranicznika podprogramu *ESM*. Ostatni element struktury – licznik wywołań – jest wypełniany i używany w trakcie konsolidacji.

Nazwa podprogramu – submodelu
Znacznik zdefiniowania
Adres początkowy
Adres końcowy
Liczba argumentów
Adres pierwszego argumentu
Adres argumentu zwrotnego
Adres pierwszej zmiennej lokalnej
Adres ostatniej zmiennej lokalnej
Licznik wywołań podprogramu

Rys. 1. Zawartość struktury *ALL_SUBROUTS* opisującej podprogram

Należy ponadto dodać, że początkowy ogranicznik podprogramu-submodelu *BSM* i nagłówek (początkowy wiersz) podprogramu wspólnie generują kod pośredni pokazany na rys. 2. W tym przykładzie liczba argumentów jest większa od czterech i generowane są wówczas dwa wiersze (lub więcej, gdy liczba argumentów jest większa od ośmiu) kodu pośredniego. Tylko w pierwszym wierszu jest umieszczana informacja o nazwie podprogramu – indeks tablicy struktur *ALL_SUBROUTS*.

	Nazwa podprogramu	BSM	Argument nr 1	Argument nr 2	Argument nr 3	Argument nr 4
	50xx	1073	4xxx	4xxx + 1	4xxx + 2	4xxx + 3

		BSM	Argument nr 5	Argument nr 6	–	–
		1073	4xxx + 4	4xxx + 5	–	–

Rys. 2. Kod pośredni nagłówka podprogramu-submodelu

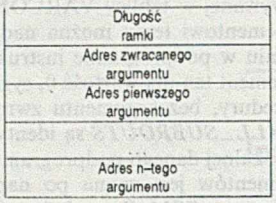
Podczas kompilacji deklaracji INP, do odpowiednich nazw argumentów formalnych w tablicy *NAMARGS* są dopisywane znaczniki sygnalizujące, że dany argument jest argumentem wejściowym. W praktycznej realizacji kompilatora tablica ta jest tablicą struktur, a pojedyncza struktura składa się z łańcucha znakowego (nazwa argumentu) i dwóch zmiennych typu całkowitego, przy czym jedna z nich jest znacznikiem wejściowym, a druga – adresem argumentu aktualnego, wyznaczanym w trakcie konsolidacji.

Zmienna kontrolna logiczna (opcjonalna)	Rezultat podprogramu (opcjonalny)	CSP – funkcja standardowa wywołania	Nazwa podprogramu	Adres listy adresów argumentów	Liczba argumentów	–
0xxx	0xxx	1072	55xx	6xxx	xxxx	–

Rys. 3. Kod pośredni generowany przez wywołanie podprogramu

Zidentyfikowane przez kompilator wywołania podprogramu powoduje wygenerowanie kodu wynikowego w postaci pokazanej na rys. 3. Identyfikacja wywołania podprogramu typu procedury takiego jak LOOP1 w podanym wcześniej przykładzie, jest dość prosta. W przypadku podprogramu typu funkcji, takiego jak LOOP2, kompilator początkowo zakłada, że napotkał zmienną, a dopiero lewy nawias otwierający

listę argumentów jest sygnałem, że chodzi tu o podprogram. Tego rodzaju sytuacja wymaga określonych modyfikacji algorytmu rozbioru wyrażeń arytmetycznych, gdyż oczywiście wywołanie funkcji może wystąpić wewnątrz wyrażenia, bez jawnego i bezpośredniego określania wartości określonej zmiennej, tak jak w podanym prostym przykładzie programu.



Rys. 4. Ramka z listą adresów argumentów aktualnych podprogramów

Nazwa wywoływanego programu jest wpisywana do tablicy *CALL_SUBR_NAME* złożonej z łańcuchów znakowych. Element *INTFRM* [] [3] kodu pośredniego (rys. 3) jest indeksem tej tablicy, przy czym jest to obiekt klasy nr 5 i podklasy (przed konsolidacją) nr 5–9, co zaznaczono na rys. 3 przez poprzedzenie indeksu „*xx*" liczbą 55. Wywołanie podprogramu, tak jak i definicja, może mieć dowolną liczbę argumentów i argumenty te są grupowane w ramki o strukturze pokazanej na rys. 4. Ramka ta zawiera informacje o swojej długości, o adresie zwracanego argumentu (indeks tablicy zmiennych globalnych i lokalnych *VARCON*) oraz o adresach argumentów aktualnych. Minimalna ramka argumentów ma w przypadku podprogramów bezargumentowych długość równą 2, gdyż nawet taki podprogram najczęściej coś zwraca instrukcją *RET*. Ramki argumentów są umieszczane w tablicy liczb całkowitych *LIST_ADDR_ARGS*. Indeks tej tablicy podający informację o położeniu początku ramki argumentów danego wywołania, jest zapisywany jako element *INTFRM* [] [4] kodu pośredniego. Informacja zawarta w *INTFRM* [] [5] ma charakter redundancyjny, gdyż liczba argumentów aktualnych może być określona jako długość ramki zmniejszona o 2.

Konsolidacja w paralelnym języku symulacyjnym

Konsolidacja w typowym języku symulacji układów dynamicznych różni się dość istotnie od konsolidacji w uniwersalnych językach algorytmicznych. Podstawową przyczyną różnic jest paralelizm języka i powiązana z tym konieczność uporządkowania instrukcji w celu osiągnięcia tzw. kolejności obliczeniowej [3, 4, 5]. Kolejność obliczeniowa oznacza, że na początek programu (właściwie kodu pośredniego) należy wysunąć instrukcje, których argumenty (lub elementy wyrażeń arytmetycznych po prawej stronie instrukcji podstawienia) są w pełni określone za pomocą deklaracji *CON* lub też są to rezultaty całkowania o znanych oczywiście warunkach początkowych. Drugą przyczyną różnic są ograniczenia wynikające z algorytmów całkowania układów równań różniczkowych.

Ze względu na niezbyt duże rozmiary programów, problem relokacji podprogramów jest rozwiązany w kompilatorze języka AMIL bardzo prosto: podprogramy-submodele są kompilowane łącznie z programem głównym. Konsolidacja, uwzględniająca wiele aspektów sortowania (kolejność obliczeniowa) jest jednak bardziej złożona. Wstępnym etapem konsolidacji jest sprawdzenie, czy argumenty podprogramów nie umieszczone na liście deklaracji *INP* są obliczane wewnątrz poszczególnych podprogramów. Operacja ta wykorzystuje znaczniki wejścia-wyjścia tablicy struktur *NAMARGS* i przegląda poszczególne podprogramy w granicach określonych przez odpowiednie elementy właściwej struktury *ALL_SUBROUTS*. Sprawdzane jest tu wystąpienie argumentów wyjściowych wśród rezultatów (element *INTFRM* [] [1] kodu pośredniego) w danym podprogramie. Operacja ta ma charakter diagnostyczny i jej negatywny wynik nie musi być równoznaczny z rzeczywistym błędem logicznym.

Właściwa operacja konsolidacji polega na zastąpieniu nazwy wywoływanego podprogramu tzn. indeksu tablicy *CALL_SUBR_NAME* znajdującego się w elemencie INTFRM [] [3] kodu pośredniego (rys. 3), indeksem wskazującym strukturę opisującą definicję danego podprogramu-submodelu (rys. 1). Odszukanie właściwej struktury opisującej definicję wywoływanego podprogramu odbywa się przez porównywanie nazwy z tablicy *CALL_SUBR_NAME* z nazwami zawartymi w strukturach *ALL_SUBROUTS*. Ta ostatnia struktura zawiera m.in. informację o początkowym adresie podprogramu, a więc miejscu, pod które należy skoczyć. Przy okazji wykonywania tej operacji jest sprawdzona zgodność liczby argumentów wywołania i definicji, a także zwracanie przez podprogram określonej wartości, jeśli wymaga tego wywołanie (różny od zera element INFRM [] [1] kodu pośredniego). Argumenty aktualne wejściowe (odpowiadające argumentom formalnym z listy deklaracji *INP*) w tablicy *LIST_ADDR_ARGS* są opatrywane znacznikami wejścia. Informacja ta jest wykorzystywana później w trakcie sortowania.

W tym miejscu należy wspomnieć o bardzo istotnej decyzji dotyczącej składni podprogramów-submodeli, a ściślej – decyzji używania określonych funkcji standardowych wewnątrz podprogramów. Decyzje te mają istotny wpływ na mechanizm kojarzenia argumentów formalnych i aktualnych w trakcie konsolidacji i później – w fazie wykonywania programu. Starsze języki symulacji układów dynamicznych [4] nie zezwalały na używanie w podprogramach standardowej funkcji całkowania. Współczesne języki tej grupy [1, 3], a m.in. również AMIL, dopuszczają używanie tej funkcji, co pozwala konstruować submodele bez jakichkolwiek istotnych ograniczeń. W starszych językach sposób używania podprogramów nie różni się od sposobu znanego użytkownikom uniwersalnych języków algorytmicznych. Podprogram-submodel jest nie tylko narzędziem dekompozycji opisu problemu, ale może być wielokrotnie używany (wywoływany) w celu wykonania tych samych lub podobnych sekwencji obliczeń. Jeśli jednak dopuszcza się używanie w podprogramie-submodelu instrukcji całkowania, to należy zauważyć, że całkowanie układu równań różniczkowych wymaga równoczesnego dostępu do wartości wszystkich zmiennych całkowanych, a tego nie zapewni, w przypadku wielokrotnego wywoływania tego samego podprogramu, nawet nadanie tym zmiennym statusu zmiennych statycznych. Najprostszym rozwiązaniem zastosowanym w systemie symulacyjnym AMIL jest wprowadzenie ograniczenia, że dany podprogram-submodel może być użyty (wywoływany) tylko jeden raz. Oczywiście, gdy dany podprogram-submodel ma być użyty (wywoływany) wielokrotnie, to wystarcza zmiana samej nazwy w nagłówku definicji i powtórzenie tekstu podprogramu. Bardziej eleganckim rozwiązaniem byłoby przechowywanie podprogramów-submodeli w bibliotece i dołączanie ich z indywidualnymi unikatowymi nazwami. Inny wariant rozwiązania, to wprowadzenie dwóch klas podprogramów-submodeli, rozróżnianych (lub nie) składnią, dopuszczających lub też nie dopuszczających używanie integratorów. Podprogramy drugiej klasy mogłyby być wywoływane wielokrotnie. Rozwiązanie przyjęte w systemie symulacyjnym AMIL jest jednym z prostszych i uwzględnia fakt, że dobry język symulacyjny powinien być zwarty i prosty. Wpływ na tę decyzję miał też fakt, że zakres zastosowań obejmuje symulację układów małych i średnich.

Przyjęcie tych założeń dotyczących podprogramów-submodeli i wynikających z nich ograniczeń spowodowało, że w trakcie konsolidacji sprawdzana jest liczba wywołań danego podprogramu i zapisywana w strukturze *ALL_SUBROUTS* (rys. 1). Przed podjęciem decyzji o wprowadzeniu integratorów do wnętrza podprogramów, było rozważane użycie stosu do przekazywania argumentów do podprogramów. Ze względu jednak na dość proste i sztywne powiązanie podprogramów i wywołań, w trakcie konsolidacji poszczególnym argumentom formalnym przypisuje się jednoznaczne adresy argumentów aktualnych, dokonując odpowiedniego zapisu w tablicy struktur *NAMARGS* w miejscu określonym przez położenie nazwy argumentu formalnego.

Sortowanie kodu pośredniego

Sortowanie kodu pośredniego jest związane z paralelizmem języków symulacji układów dynamicznych. Program w takim języku (część modelowa [3]) nie tyle opisuje określoną sekwencję operacji, ile określa zależności między wielkościami (zmiennymi) w mniej lub bardziej złożonym obiekcie fizycznym. Jak już wspominano, celem sortowania jest doprowadzenie kolejności tych operacji do porządku obliczeniowego. Na początku zakłada się, że znane są tylko wartości zmiennych zdefiniowane za pomocą deklaracji CON oraz zmienne będące wynikami całkowania (w poprawnym programie muszą być określone warunki początkowe). Kolejne kroki sortowania polegają na odszukaniu w kodzie pośrednim tych wierszy (operacji elementarnych), w których wszystkie argumenty są określone oraz oznaczenie rezultatów tych operacji jako zdefiniowanych.

Sortowanie podprogramów-submodeli jest traktowane bardzo różnie. W niektórych językach symulacyjnych (*MIMIC* [4]) podprogramy są sortowane, a w innych (*CSMP* [1, 2]) prowadzi się sortowanie podprogramów w pełnym zakresie. W języku AMIL przyjęto zasadę pośrednią – jest sprawdzana w podprogramach, przy złagodzonych kryteriach, kolejność obliczeniowa, bez prowadzenia sortowania. Złagodzenie kryteriów sortowania polega na dopuszczeniu tzw. podwójnego definiowania zmiennej, co może mieć postać:

$VARI = X2 + 2{,}0$
$VARI = VARI + X3$

Taka sekwencja operacji w programie głównym zostałaby zdiagnozowana jako błędna (podwójne zdefiniowanie *VAR*1, nie akceptowane wobec paralelizmu języka). W programie głównym należałoby więc użyć równoważnej instrukcji:

$VAR1 = X2 + 2{,}0 + X3$

Do sortowania podprogramu-submodelu, a właściwie sprawdzenia kolejności obliczeniowej, dochodzi w przypadku napotkania jego wywołania. Pierwszym krokiem jest sprawdzenie, czy wszystkie aktualne argumenty wejściowe (z listy INP) są już obliczone lub zdefiniowane przed wywołaniem. Informacje o adresach argumentów są pobierane z tablicy struktur *LIST_ADDR_ARGS*, gdzie znajdują się również znaczniki wejścia-wyjścia. Jeśli wynik tego sprawdzenia jest pozytywny, można przejść do sortowania wnętrza podprogramu, a jeśli nie, to zgodnie z zasadami sortowania wywołanie podprogramu należy przenieść do bloku operacji niesortowanych i zająć się następną operacją elementarną z kodu pośredniego.

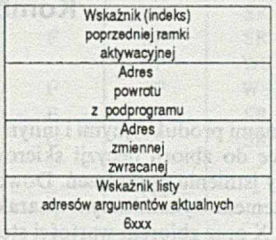
Sortowanie wnętrza podprogramu-submodelu, po uwzględnieniu omówionego wcześniej złagodzenia kryteriów, przebiega w taki sposób, że kolejne zdefiniowane (obliczone) zmienne są opatrywane znacznikiem tzw. tymczasowego zdefiniowania, jeśli nie były już poprzednio zdefiniowane (obliczone). Gdy wszystkie zmienne globalne i lokalne w podprogramie są zdefiniowane lub zdefiniowane tymczasowo, to podprogram ma właściwą kolejność obliczeniową. Znaczniki zdefiniowania tymczasowego są zamieniane na normalne znaczniki zdefiniowania. W przeciwnym przypadku, gdy kolejność obliczeniowa jest niewłaściwa, znaczniki zdefiniowania tymczasowego są zastępowane znacznikami niezdefiniowania, powraca się do sortowania operacji występujących po wywołaniu, a podprogram-submodel może okazać się sortowalny nieco później, gdy zostanie obliczona (wysunięta wyżej w kodzie pośrednim wskutek sortowania) jakaś nowa zmienna globalna.

Dość subtelny problem wyłania się w przypadku występowania integratorów w podprogramach, takich jak np. podprogramy LOOP1 i LOOP2 w podanym na wstępie przykładzie. Otóż opisany algorytm już na wstępie stwierdziłby, że zmienna całkowana *edif* jest niezdefiniowana, a więc podprogram nie ma właściwej kolejności obliczeniowej. Byłaby to jednak przedwczesna dyskwalifikacja, gdyż oczywiście możliwe jest określenie rezultatu całkowania (wartość początkowa) i kontynuowanie obliczeń w obrębie podprogramu. Aby rozwiązać ten problem, niezdefiniowane zmienne całkowane traktuje się w sposób specjalny – po prostu zlicza się je i zapisuje ich adresy w roboczej tablicy adresów niezdefiniowanych zmiennych podcałkowych, bez diagnozowania takich sytuacji jako błędnych. W przypadku każdego rezultatu obliczanego wewnątrz podprogramu sprawdza się, czy zmienna ta nie znalazła się w roboczej tablicy niezdefiniowanych zmiennych podcałkowych i w przypadku pozytywnego rezultatu testu – usuwa z tej tablicy. W poprawnym, w sensie kolejności obliczeniowej podprogramie, tablica ta powinna być pusta na końcu operacji sortowania podprogramu.

Wykonanie podprogramów

Wykonywanie podprogramów w systemie symulacyjnym AMIL jest realizowane przez interpreter [5] wykorzystujący kod pośredni, zawarty w tablicy *INTFRM*, oraz pewne dodatkowe informacje. Wprowadzenie mechanizmu definiowania i używania podprogramów wymagało rozbudowania tych dodatkowych informacji o elementy służące do budowy ramek aktywacyjnych podprogramów oraz do druku zrzutu post-mortem programu w przypadku błędu wykonania. Te dodatkowe informacje to:

- długość kodu pośredniego programu głównego,
- liczba wywołań podprogramów,
- długość efektywna tablicy struktur *LIST_ADDR_ARGS* z ramkami argumentów aktualnych (rys. 4),
- części adresowe (rys. 4) struktur *LIST_ADDR_ARGS* (w fazie interpretacji niezbędny jest tylko jeden element tej struktury),
- liczba argumentów formalnych w definicjach podprogramów-submodeli, tzn. długość efektywna tablicy struktur *NAMARGS*,
- tablica struktur *NAMARGS* z nazwami argumentów formalnych oraz ich adresami aktualnymi, wyznaczonymi w trakcie konsolidacji (te dwa elementy struktury wystarczają interpreterowi),
- liczba zdefiniowanych podprogramów, tzn. efektywna liczba struktur w tablicy *ALL_SUBROUTS*,
- nazwy i adresy początkowe podprogramów (tylko te dwa elementy) z poszczególnych struktur *ALL_SUBROUTS*.



Rys. 5. Zawartość ramki aktywacyjnej podprogramu

Ramka aktywacyjna podprogramu zawiera cztery elementy (rys. 5). Bliższa analiza zawartości ramki aktywacyjnej wykazuje, że czwarty element tej ramki ma w istocie charakter redundancyjny, ze względu na opisane wcześniej sztywne i jednoznaczne powiązanie wywołań i definicji podprogramów. Wskaźnik ramki adresów argumentów aktualnych w tablicy *LIST_ADDR_ARGS* pozostawiono ze względu na ewentualną perspektywę wprowadzenia drugiej klasy podprogramów – bez operacji całkowania. Informacja o adresie powrotnym nie ma wprawdzie charakteru redundancyjnego, ale mogłaby być, z podobnych powodów, w sposób sztywny powiązana z danym podprogramem-submodelem. Nie zdecydowano się na takie rozwiązanie, aby pozostawić otwartą drogę dla ewentualnych przyszłych modyfikacji składni języka AMIL. Ramki aktywacyjne są umieszczane na stosie wywołań, co pozwala wykonywać zagnieżdżone podprogramy.

* * *

System symulacyjny AMIL po wprowadzeniu opisanego tu mechanizmu, pozwalającego na korzystanie z podprogramów-submodeli, nie ustępuje w istotny sposób współczesnym językom i systemom symulacji układów dynamicznych [1, 3]. Jego istotną zaletą jest przyjazne dla użytkownika zintegrowane środowisko, a także wprowadzona ostatnio interakcyjna grafika [6]. W najbliższym czasie przewiduje się zmodyfikowanie składni zapisu równań różniczkowych w sposób odpowiadający współczesnym standardom.

Przedstawione tu rozwiązania programowe nie są ani jedyne, ani wzorcowe. W kilku przypadkach opisano przyczyny odstępstw od rozwiązań bardziej eleganckich i uniwersalnych. Choć przedmiotem prezentacji jest wnętrze kompilatora i konsolidatora języka paralelnego, wiele mechanizmów może znaleźć zastosowanie i w innych przypadkach.

LITERATURA

[1] Crosbie R.E., Javey S., Hay J.L., Pearce J.G.: ESL – A new Continuous System Simulation Language. SIMULATION, Vol. 44, No. 5, May 1985, pp: 242–246

[2] Glisson T.H.: Introduction to System Analysys. McGraw Hill, New York 1985

[3] Ketennis D.L.: COSMOS – A Simulation Language for Continuous, Discrete and Combined Models. SIMULATION, Vol. 58, No 1, January 1992, pp. 32–34

[4] Sansom F.J., Peterson H.E.: MIMIC Programming Manual. Technical Report SEG-TR-67-31, Wright-Patterson AFB, 1967

[5] Stabrowski M.: AMIL – interakcyjny język symulacji układów dynamicznych. INFORMATYKA 1992, nr 11, s. 1–4

[6] Stabrowski M.: Nowe narzędzia w systemie symulacyjnym AMIL – interakcyjna grafika i podprogramy. Materiały Sympozjum „Modelowanie i Symulacja Systemów Pomiarowych". Wydawnictwo AGH, Kraków 1993.

Modele zbiorów rozmytych w analizie decyzyjnej przedsiębiorstwa Część 2

Tadeusz Witkowski
Kombinat Przemysłu Narzędziowego VIS
Warszawa

Zarządzanie systemami produkcyjnymi i innymi organizacjami sprowadza się w praktyce do zbioru decyzji skierowanych na realizację określonego celu przy istnieniu ograniczeń. Dowolny proces zarządzania (sterowania) systemem dynamicznym charakteryzuje się zbiorem stanów tego systemu X oraz zbiorem wartości sterowań U, gdzie stany i wartości sterowań dla momentu czasu $t \in T$ oznaczamy odpowiednio przez x_t i u_t. Funkcjonowanie systemu, tj. przejścia z jednego stanu do innego stanu pod wpływem sterowania, są określane przez równanie stanu: $x_{t+1} = f(x_t, u_t)$. Zakłada się, że zmiany stanów odbywają się w dyskretne momenty $T = \{t\} = \{0, 1, 2, ...\}$. Jeżeli oprócz tego system ma skończoną liczbę stanów i sterowań, to wieloetapowy proces podejmowania decyzji można przedstawić za pomocą automatu (X, U, f, x_o, X_t), gdzie $X = \{x_1, ..., x_N\}$, $U = \{u_1, ..., u_m\}$, f – funkcja określająca przejście z jednego stanu do drugiego, x_o – stan początkowy, $X_t \subset X$ – zbiór stanów końcowych.

Wieloetapowe procesy podejmowania decyzji

Dynamiczny problem podejmowania decyzji można przedstawić także za pomocą grafu $G = \{X, V\}$, gdzie X – zbiór wierzchołków odpowiadających zbiorowi stanów systemu, a $V \subset X \times X$ – zbiór ukierunkowanych łuków (x_j, x_i), opisujących przejścia ze stanu j do stanu i pod wpływem sterowania u_t. Dowolną kolejność sterowań za pomocą których system przechodzi ze stanu początkowego do końcowego (utożsamianego z celem), nazywamy strategią, a dowolną kolejność sterowań przedstawiającą część zbioru tych sterowań nazywamy strategią częściową.

W przypadku systemu dynamicznego z rozmytymi celami i ograniczeniami funkcjonowania, przy analizie podejmowania decyzji stosuje się metodę programowania dynamicznego (zasadę optymalności Bellmana) oraz metodę podziału i oszacowań. W tym przypadku równanie stanu przyjmuje postać $\mu_{x_{t+1}} = f(\mu_{x_t}, u_t)$, $t = 1, ..., N$, gdzie $\mu_{x_{t+1}}$, μ_{x_t}: $x \to [0, 1]$ – stany rozmyte systemu dynamicznego odpowiednio w momentach czasu $t + 1$ i t [2].

Przykład 5. Rozpatrzmy zastosowanie rozmytego programu dynamicznego do wyboru strategii rozwoju mocy produkcyjnych przedsiębiorstwa zgodnie z warunkami gry decyzyjnej, w której decyzje podejmuje się kilka razy w trakcie jej realizacji. Załóżmy, że zbiory możliwych decyzji (sterowań) i możliwych stanów są następujące: $U = \{u_i\}$, $i \in 1,3$; $X = \{x_i\}$, $i \in 1,5$. Ewolucja systemu opisana jest za pomocą funkcji: $x_{i+1} = f(x_i, u_i)$.

Tabela 1. Stopnie pożądanego stanu systemu i możliwości zastosowania sterowania

Stan x_i	Stopień pożądanego stanu systemu	Stopień możliwości zastosowania sterowania (decyzji) u_i na określonym etapie		
		u_1	u_2	u_3
x_1	0,25	1,00	1,00	0,70
x_2	1,00	1,00	0,60	0,40
x_3	0,70	1,00	0,90	0,30
x_4	0,60	1,00	0,90	0,60
x_5	0,40	1,00	0,80	0,40

Przyjmujemy, że system początkowo znajduje się w stanie x_1 i po $N = 5$ etapach oczekujemy, że znajdzie się w stanie $x_5 = x_n$. Zakładamy, że celem jest osiągnięcie przez system stanu, opisanego w postaci następującego rozmytego podzbioru $G = x_1/\mu_1 + x_2/\mu_2 + x_3/\mu_3 + x_4/\mu_4 + x_5/\mu_5$, gdzie $\mu_i \in 0, 1$. Osiągnięcie tego celu jest możliwe za pomocą zastosowania odpowiednich sterowań na kolejnych etapach procesu decyzyjnego, przy czym decyzje nie mogą być wybierane dowolnie, lecz powinny spełniać określone ograniczenia, przedstawione w postaci rozmytych podzbiorów. Stopnie pożądanego stanu systemu i możliwości zastosowania sterowania zawiera tabela 1.

Niech C_i oznacza ograniczenie, które powinno być spełnione na $(i + 1)$-tym etapie, przy czym następującą postać mają:

$C^0 = u_1/\mu_{C_1}^0 + ... + u_3/\mu_{C_3}^0$; $C^1 = u_1/\mu_{C_1}^1 + ... + u_3/\mu_{C_3}^1$;
$C^2 = u_1/\mu_{C_1}^2 + ... + u_3/\mu_{C_3}^2$; analogicznie określa się C^3 i C^4, gdzie $\mu_{C_i}^0$, $\mu_{C_i}^1$, $\mu_{C_i}^2$, $\mu_{C_i}^3$, $\mu_{C_i}^4$ mają określone wartości z przedziału [0,1].

Rozwiązaniem D jest osiągnięcie celu oraz spełnienie pięciu przedstawionych ograniczeń. Ponieważ cel i ograniczenia przedstawione są w postaci zbiorów rozmytych, decyzja jest też zbiorem rozmytym: $D = G \cap C^0 \cap C^1 \cap C^2 \cap C^3 \cap C^4$. Zbiory rozmyte G i C^i określa się odpowiednio za pomocą funkcji przynależności μ_G oraz μ_C^i Wartość

Tabela 2. Określenie stanu systemu i stopnia usatysfakcjonowania z zastosowania sterowania

Zastosowana strategia zbiór decyzji	Osiągany stan systemu x_i	Stopień usatysfakcjonowania
$u_1\ u_1\ u_1$	x_1	0,25
$u_1\ u_1\ u_2$	x_2	0,90
$u_1\ u_1\ u_3$	x_4	0,30
$u_1\ u_2\ u_1$	x_2	0,60
$u_1\ u_2\ u_2$	x_3	0,60
$u_1\ u_2\ u_3$	x_1	0,25
$u_1\ u_3\ u_1$	x_4	0,40
$u_1\ u_3\ u_2$	x_4	0,40
$u_1\ u_3\ u_3$	x_2	0,30
$u_2\ u_1\ u_1$	x_2	1,00
$u_2\ u_1\ u_2$	x_3	0,70
$u_2\ u_1\ u_3$	x_1	0,25
$u_2\ u_2\ u_1$	x_3	0,60
$u_2\ u_2\ u_2$	x_5	0,40
$u_2\ u_2\ u_3$	x_3	0,30
$u_2\ u_3\ u_1$	x_1	0,25
$u_2\ u_3\ u_2$	x_2	0,40
$u_2\ u_3\ u_3$	x_4	0,30
$u_3\ u_1\ u_1$	x_4	0,60
$u_3\ u_1\ u_2$	x_4	0,60
$u_3\ u_1\ u_3$	x_2	0,30
$u_3\ u_2\ u_1$	x_4	0,60
$u_3\ u_2\ u_2$	x_4	0,60
$u_3\ u_2\ u_3$	x_2	0,30
$u_3\ u_3\ u_1$	x_2	0,40
$u_3\ u_3\ u_2$	x_3	0,40
$u_3\ u_3\ u_3$	x_1	0,25

$\mu_G(x)$ interpretuje się jako stopień aspiracji [2] lub stopień pożądanego stanu x, a wartość $\mu_C^i(u)$ jako stopień możliwości zastosowania sterowania u na i-tym etapie. Wartości funkcji μ_D, przedstawiające połączenie stopni pożądanego stanu ze stopniami możliwości zastosowania sterowania, nazywamy stopniem usatysfakcjonowania.

Funkcje μ_G i μ_{C_i} są określone na różnych zbiorach, przy czym:
$\mu_G: \{x_1, x_2, x_3, x_4, x_5\} \rightarrow [0,1]$, a $\mu_{C_i}: \{u_1, u_2, u_3\} \rightarrow [0,1]$.
Stopień usatysfakcjonowania, np. przy zastosowaniu sterowania u_1, u_2 u_3, tj. takim, przy którym na początku przyjmuje się decyzję u_1, potem u_2, a na końcu u_3, określa się następująco:
$\mu_D(u_1, u_2, u_3) = \mu_G \wedge \mu_C^0 \wedge \mu_C^1 \wedge \mu_C^2 \wedge \mu_C^3 \wedge \mu_C^4$

Na podstawie tabeli 2 można zauważyć, że przy sterowaniu u_1, u_2, u_3 system nadal pozostaje w stanie x_1, a stopień usatysfakcjonowania takim wynikiem jest dość niski i wynosi 0,25. W tabeli tej przedstawiono stopnie usatysfakcjonowania otrzymanymi wynikami przy zastosowaniu różnorodnych strategii. Jak widać, tylko przy zastosowaniu strategii decyzyjnej u_2, u_2, u_2 system może przejść ze stanu x_1 do stanu x_5, a poziom usatysfakcjonowania μ_D z takiej strategii będzie równy 0,4. Tak więc, mimo osiągnięcia przez system pożądanego stanu, stopień μ_D jest również stosunkowo niski, co można interpretować np. dużymi kosztami związanymi z osiągnięciem takiego stanu.

Konstrukcja reguł wnioskowania za pomocą algorytmu indukcyjnego

Dla pozyskiwania wiedzy przy projektowaniu systemu eksperckiego dla rozwiązywanego problemu określa się zestaw przykładów wnioskowania rozmytego na określonym zbiorze sytuacji. Zestaw ten zawiera zbiór cech określających sposób wyboru alternatywnych zachowań w danej sytuacji. Każdy przykład zawiera rozmyte wartości cech oraz opis reguł, którymi należy kierować się w przypadku wystąpienia określonej kombinacji poszczególnych wartości cech.

Dostatecznym warunkiem otrzymania prawidłowego wniosku (1) jest to, aby określone wartości każdej cechy opisującej daną sytuację odnosiły się zawsze do jednej tylko wartości wyniku wnioskowania rozmytego.

Schemat wnioskowania użytego algorytmu indukcyjnego [1] jest następujący. Najpierw określa się jedną z cech jako bazową, a następnie dla każdej wartości tej cechy wypisuje się numery wierszy tabeli przedstawiającej sposób wnioskowania w danej sytuacji, dla których wartości są jednakowe i sprawdza się spełnienie równości prawej części warunku:

$$R^1 \forall t_i, t_j (t_i(X) = t_j(X) \Rightarrow t_i(WR) = t_j(WR)),\ i \neq j \qquad (1)$$

gdzie: t_j oznacza wiersz tabeli określającej sposób wnioskowania, $j \in 1, k$ numer wiersza przy czym każdy wiersz jest scharakteryzowany za pomocą lingwistycznych wartości cech A_i określających sytuację, X – podzbiór cech $X \notin ZM$, $ZM = \{A_1, ..., A_r\}$, $t_j(X)$ – zestaw wartości cech zbioru X dla wiersza t_j, $t_j(WR)$ – wartość wyniku wnioskowania rozmytego dla wiersza t_j tabeli.

Jeżeli implikacja (1) jest spełniona, to formułuje się ostateczną hipotezę. W przeciwnym przypadku generowana jest hipoteza niepełna oraz przeprowadzana redukcja tabeli (wykreślane są wiersze, dla których jest spełniona równość oraz kolumna charakteryzująca cechę bazową w danym momencie analizy decyzyjnej). Następnie rozpatrywana jest kolejna cecha. Analogiczne operacje są przeprowadzane dla następnych cech. Końcowa hipoteza jest formułowana z uwzględnieniem niepełnej hipotezy, otrzymanej w poprzednim kroku algorytmu indukcyjnego.

Implikacja (1) przedstawia kryterium, na podstawie którego jest konstruowane drzewo decyzyjne. Jako najlepsze kryterium wyboru cechy bazowej w każdym kroku algorytmu przyjmuje się implikację:

$$\forall t_i, t_j (t_i(WR) \neq t_j(WR) \Rightarrow t_i(X) \neq t_j(X)),\ i \neq j \qquad (2)$$

Ze wszystkich możliwych cech najpierw wybiera się tę, która ma maksymalną liczbę par wierszy w tabeli opisującej sytuacje przybliżonego wnioskowania, w których przy różnych wartościach wniosku WR, różne są także wartości cech tworzących zbiór X.

Przykład 6. Dla rozpatrywanego problemu decyzyjnego (określenia prawdopodobieństwa sukcesu przedsiębiorstwa w przetargu) przybliżone wnioskowanie przeprowadza się z uwzględnieniem następujących czterech cech [4]:

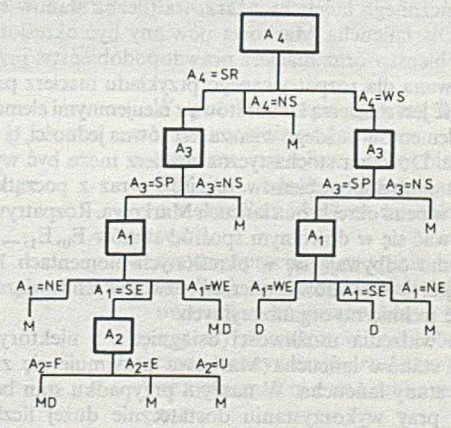
- kosztów realizacji produkcji zamówień oferowanych w przetargu,
- wysokości narzutu wpływającego na wielkość zysku,
- warunków modelowania różnych liczb losowych mających np. wpływ na koszty rzeczywiste,
- sytuacji rynkowej, określonej za pomocą relacji n/m. Przy czym m – liczba przedsiębiorstw, a n – liczba zamówień oferowanych w przetargu, realizowanym w postaci gry decyzyjnej.

Ostatecznie jest konstruowana reguła wnioskowania w postaci drzewa decyzyjnego typu ⟨IF...THEN...ELSE...⟩.

Tabela 3. Zestaw reguł opisujących sposób podejmowania decyzji o prawdopodobieństwie sukcesu w przetargu

Nr wiersza	Koszty produkcji	Warunki gry	Sytuacja rynkowa	Wartość narzutu	Prawdopodobieństwo sukcesu w przetargu
1	WE	–	SP	NS	D
2	SE	F	SP	NS	D
3	WE	–	SP	SR	MD
4	SE	F	SP	SR	MD
5	WE	–	SP	WS	M
6	SE	F	SP	WS	M
7	SE	E	–	SR	M
8	SE	U	–	SR	M
9	NE	–	–	–	M
10	–	–	NS	–	M

W celu przeprowadzenia formalnego rozumowania, wprowadza się następujące oznaczenia: A_1 – koszty produkcyjne, A_2 – warunki gry, określane przez Centrum Gry, A_3 – sytuacja rynkowa, A_4 – wysokość narzutu, PROBSU – prawdopodobieństwo sukcesu w przetargu. Przyjmujemy określone wartości lingwistyczne dla poszczególnych cech: dla A_1 – wysokie WE, średnie SE, niskie NE, dla A_2 – sprzyjające F, niesprzyjające E i umiarkowane U, dla A_3 – sprzyjająca SP i niesprzyjająca NS, dla A_4 – wysoki WS, średni SR i niski NS, a dla PROBSU – duże – D, średnie – MD i małe – M. Ponadto istnieje sytuacja określona symbolem „–", dla której możliwe jest wystąpienie dowolnej wartości rozpatrywanej cechy. Zestaw reguł opisujących sposób podejmowania decyzji o prawdopodobieństwie sukcesu w przetargu wraz z lingwistycznymi wartościami cech i wyniku wnioskowania, przedstawiono w tabeli 3.



Kompleksowa reguła wnioskowania dla rozpatrywanej sytuacji problemowej

Kompleksową regułę w postaci drzewa decyzyjnego przedstawiono na rysunku.

Badanie efektywności decyzji z zastosowaniem łańcuchów Markowa

Zachowanie się systemów dynamicznych i ich badanie opiera się obecnie głównie na podejściu wykorzystującym oprócz wielkości wyjściowych i wejściowych, również zmienne stanu. Znając aktualne wartości zmiennych stanu oraz aktualne wartości zmiennych wejściowych można przewidzieć przyszłe zachowanie się obiektu, niezależnie od tego, co działo się z systemem w przeszłości.

Przy wyborze decyzji dotyczącej zarządzania przedsiębiorstwem dąży się do tego, aby w rozpatrywanych warunkach dla określonych ograni-

czeń produkcyjnych i finansowych ustalić najbardziej efektywną strategię. Przy takim podejściu można określić spośród całego zbioru badanych wariantów takie zestawy decyzji, które w istniejących warunkach są najbardziej racjonalne. W ten sposób analizowane są problemy dynamiki zbliżenia do optymalnego rozwiązania ciągu decyzji, konstruowanego z wykorzystaniem zmiennych wejściowych, charakteryzujących np. zamierzenia rozwojowe przedsiębiorstwa. Przeprowadzenie w tym zakresie badań jest możliwe na podstawie przetwarzania danych statystycznych, charakteryzujących zmiany wartości kryterium lub wektora kryteriów dla różnych wariantów decyzji.

Formułowanie takich danych statystycznych odbywa się przez wielokrotną konstrukcję zamierzeń rozwojowych za pomocą odpowiedniego algorytmu symulacyjnego. Wykorzystując za pomocą modelowania symulacyjnego wybrane w określony sposób zestawy decyzji, otrzymujemy ciąg stanów systemu. Algorytm symulacyjny przy konstrukcji poszczególnych strategii rozwoju oraz przejściu od jednego wariantu do drugiego pozwala zmieniać zarówno zmienne określające zamierzenia rozwojowe, jak i ich wartości, a także przeprowadzać różne schematy zapamiętywania poszczególnych konstrukcji strategii. Dla strategii odpowiadających wariantom zamierzeń rozwojowych, istotne jest określenie charakteru przybliżenia poszczególnych strategii do optymalnego rozwiązania, lub jego granicy – dolnej w problemach minimalizacji lub górnej w problemach maksymalizacji.

W artykule rozpatrzono wartości funkcji kryterialnej f(G) z uwzględnieniem ciągu decyzji konkurujących przedsiębiorstw opisujących wpływ otoczenia w zależności od ciągu decyzji podejmowanych przez przedsiębiorstwo. W tym celu cały zakres wartości funkcji kryterialnej f(G) podzielono na przedziały. Przyjęto, że rozpatrywany system znajduje się w stanie E_i, jeżeli f(G) – wartość funkcji kryterialnej dla wariantu strategii jest zawarta w *i*-tym przedziale.

Jako kryterium oceny w obliczeniach przyjęto sumaryczny zysk w całym okresie zarządzania przedsiębiorstwem. Cały zakres wartości funkcji kryterialnej podzielono na przedziały ξ w następujący sposób. Każdy przedział ξ przedstawiono w postaci odcinka z granicami a_i oraz b_i ($a_i \leqslant \xi < b_i$). Fakt otrzymania określonej wartości f(G) z *i*-tego przedziału nazywamy przejściem systemu do stanu E_i (numer stanu jest równy numerowi przedziału, gdzie *v* – liczba przedziałów zakresu wartości funkcji f(G)). Załóżmy, że proces konstrukcji wariantów strategii jest realizowany w taki sposób, że można go przedstawić za pomocą skończonego łańcucha Markowa (liczba stanów systemu jest określona). Dla łańcucha Markowa powinny być określone rozkłady prawdopodobieństw oraz macierz prawdopodobieństw przejść.

Skonstruowana dla rozpatrywanego przykładu macierz prawdopodobieństw przejść jest macierzą kwadratową z nieujemnymi elementami, przy czym suma elementów każdego wiersza jest równa jedności, tj. macierz jest stochastyczna. Dowolna stochastyczna macierz może być wykorzystana jako macierz prawdopodobieństw przejść i wraz z początkowym rozkładem p(O) w pełni określa ona łańcuch Markowa. Rozpatrywany system może znajdować się w dowolnym spośród stanów $E_0, E_1, ..., E_n$. Zmiany stanów systemu odbywają się w określonych momentach 1, 2, ..., n pod wpływem wejść, tj. wariantów zamierzeń rozwojowych i związanych z nimi przedsięwzięć techniczno-organizacyjnych.

Z punktu widzenia możliwości osiągnięcia z niektórych stanów, określonych stanów łańcucha Markowa przyjmuje się zwykle niżej wymienione stany łańcucha. W naszym przypadku stan bezpowrotny oznacza, że przy wykorzystaniu dostatecznie dużej liczby strategii istnieje bardzo małe prawdopodobieństwo otrzymania wartości kryterialnej z *i*-tego przedziału pod warunkiem, że wyjściowym był stan oceniany na podstawie wartości kryterium z *i*-tego przedziału. Stan powrotny (rekurencyjny) oznacza, że w wyniku zastosowania dość długiej serii strategii system pozostanie w podzbiorze określonych stanów. Stan pochłaniający (absorbcyjny) oznacza, że przy realizacji określonych strategii z wysokim stopniem prawdopodobieństwa, gwarantowane jest otrzymanie wyników z wartością funkcji kryterialnej tylko z *i*-tego przedziału.

Badanie możliwości osiągnięcia określonych stanów jest w analizie decyzyjnej przedsiębiorstwa problemem bardzo ważnym. Jest to szczególnie istotne na etapie określania strategii rozwojowej przedsiębiorstwa, gdyż np. uzyskanie określonego poziomu dochodów daje możliwość realizacji założonych zamierzeń rozwojowych wraz z określeniem prawdopodobieństwa osiągnięcia takiego stanu. Badania takie, np. w postaci gier modelowych i decyzyjnych, powinny uwzględniać strategie konkurentów (wpływ otoczenia). Mają one bardzo duży wpływ na osiągnięcie zakładanych wartości funkcji kryterialnej, na podstawie których oceniana jest działalność produkcyjna przedsiębiorstwa.

Modelowanie symulacyjne i analiza wyników

Do oceny podejmowania decyzji wieloetapowych w rozpatrywanej grze decyzyjnej służy stopień zgodności rzeczywistych i prognozowanych wyników działalności finansowo-produkcyjnej przedsiębiorstw. Ponadto do oceny podejmowanych decyzji w przedstawionej sytuacji problemowej wykorzystano następujące kryteria, przedstawiające w uproszczeniu jej specyfikę, a mianowicie:

- skuteczność (efektywność) – jako stopień osiągnięcia przez system postawionych celów (porównanie tego, co zamierzono, z tym, co faktycznie osiągnięto). Wskaźnikiem tego kryterium jest w tym przypadku współczynnik sukcesu w przetargu (stosunek liczby zamówień otrzymanych do liczby zamówień przedstawionych),
- zyskowność (rentowność) – jako stosunek dochodów całkowitych do sumarycznych kosztów,
- ekonomiczność – jako stosunek zasobów wykorzystanych do zasobów faktycznie posiadanych (w tym przypadku rozpatrywane jest wykorzystanie mocy produkcyjnej),
- produktywność – jako stosunek wartości realizowanej produkcji (zamówień) do kosztów jej realizacji.

Do wielokryterialnego podejmowania decyzji (w naszym przypadku oceny działalności przedsiębiorstwa) wykorzystano podejście Jagera [2]. Zaletą tego podejścia jest to, że związane jest ono z modelem podejmowania decyzji w rozmytym otoczeniu w ujęciu Bellmana i Zadeha oraz łatwo ujmuje różne poziomy ważności poszczególnych kryteriów.

Zadanie polega na znalezieniu takiej decyzji ze zbioru $A = \{a_1, a_2, ..., a_n\}$, która najlepiej spełnia wszystkie kryteria $K = \{K_1, K_2, ..., K_m\}$, gdzie $K_k: A \to [0, 1]$, $k = 1, 2, ..., m$ wprowadza uporządkowanie decyzji ze zbioru A względem kryterium K_k. Przy poszukiwaniu najlepszej decyzji tworzy się decyzję rozmytą, np. typu minimum (tzn. $D = K_1 \cap K_2 \cap ... \cap K_m$), a następnie poszczególnym kryteriom przypisuje się wagi, odpowiednio $w_1, w_2, ..., w_m$, przy czym do ich wyznaczenia wykorzystano metodę opracowaną przez Saaty'ego, a opartą na porównaniu kolejnych par kryteriów. Wyniki jednej z gier decyzyjnych i ich wielokryterialną ocenę przedstawiono w tab. 4.

Tabela 4. Ocena działalności przedsiębiorstw na podstawie wybranych kryteriów i wskaźników

Wskaźniki i kryteria oceny	Przedsiębiorstwo						
	1	2	3	4	5	6	7
Liczba ofert na zamówienia B	19	22	29	18	20	26	29
Liczba otrzymanych zamówień A	6	11	14	7	10	13	13
Wysokość narzutu	9,4	9,3	10,6	9,8	9,35	9,09	10,1
Wartość realizowanych zamówień w jednostkach pieniężnych	1114	6654	3613	1148	1139	6839	2616
Wartość zysku przedsiębiorstwa w jednostkach pieniężnych	45	176	117	71	57	128	63
Wartość współczynnika sukcesu w przetargu A/B	0,315	0,5	0,48	0,39	0,5	0,5	0,33
Ekonomiczność	0,28	0,88	0,825	0,29	0,285	0,93	0,64
Produktywność	1,005	1,054	1,095	1,010	1,00	1,074	1,080
Preferencyjne uporządkowanie przedsiębiorstw na podstawie wielokryterialnej oceny	7	2	3	5	6	1	4

Wykorzystując określone wartości lingwistyczne (tab. 5) dla wartości wybranych kryteriów i wskaźników oceny działalności przedsiębiorstw, możemy przy projektowaniu systemów eksperckich dla danego problemu określać zestawy przykładów przybliżonego wnioskowania i na

dokończenie na s. 27

SYSTEMY CZASU RZECZYWISTEGO

REDAGUJE ZESPÓŁ: dr inż. Zbigniew FRYŹLEWICZ (Politechnika Wrocławska), prof. dr hab. Janusz GÓRSKI (Francusko-Polska Wyższa Szkoła Nowych Technik Informatyczno-Komunikacyjnych w Poznaniu), prof. dr hab. inż. Zbigniew HUZAR (Politechnika Wrocławska), dr inż. Krzysztof SACHA (Politechnika Warszawska), mgr inż. Zdzisław ŻURAKOWSKI (Fabryka Wagonów PAFAWAG we Wrocławiu) – przewodniczący zespołu.

ROK I ● Wrzesień ● 1994 ● 1

Ukazujące się obecnie publikacje na temat komputerowych systemów czasu rzeczywistego nie mają charakteru działalności systematycznej w żadnym czasopiśmie krajowym. Sporadycznie pojawiają się na ten temat publikacje jedynie w czasopismach: *Informatyka, Elektronizacja, Pomiary Automatyka Kontrola*. W tej sytuacji wkładka do *Informatyki*, jako jedyne w kraju wydawnictwo poświęcone systemom czasu rzeczywistego, zakresem tematycznym obejmować będzie wszystkie zagadnienia związane z projektowaniem i eksploatacją tego rodzaju systemów. Ponieważ część sprzętowa takich systemów najczęściej opiera się na stosunkowo tanich i łatwo dostępnych urządzeniach seryjnych wysokiej jakości, główny nacisk zostanie położony na problemy oprogramowania.

Szczególnie ważne zagadnienia, które chcielibyśmy prezentować na łamach wkładki to:

- Niezawodność i bezpieczeństwo systemów.
- Niezawodność oprogramowania.
- Specyfikacja wymagań dla systemów i oprogramowania.
- Metody i narzędzia projektowania oprogramowania.
- Organizacja procesu projektowania i kierowania projektem oraz jego dokumentacja.
- Testowanie, atestacja i certyfikowanie oprogramowania oraz systemów. Współpraca Polski w tym zakresie z państwami EWG.
- Eksploatacja oprogramowania i systemów.
- Omówienie i ocena produkowanych seryjnie elementów sprzętowych systemu.
- Normalizacja i uregulowania prawne.
- Omówienie i analiza systemów czasu rzeczywistego zrealizowanych (bądź realizowanych) w kraju i na świecie.

Oczekujemy artykułów lub komunikatów opartych na pracach własnych (badaniach naukowych, zrealizowanych projektach) oraz artykułów przeglądowych. Za szczególnie cenne uważamy artykuły prezentujące zastosowania praktyczne.

Materiały prosimy przesyłać pod adresem:
mgr inż. Zdzisław Żurakowski
Samodzielna Pracownia Pojazdów Systemów Komputerowych
F.W. PAFAWAG
ul. Fabryczna 12, 53-609 Wrocław

Systemy operacyjne czasu rzeczywistego

Adam Mrózek
Ryszard Winiarczyk
Instytut Informatyki Teoretycznej
i Stosowanej PAN
Gliwice

Pojęcie systemu operacyjnego jest w informatyce dobrze ugruntowane [11]. System operacyjny jest programem, którego podstawowym zadaniem jest udostępnianie użytkownikowi zasobów sprzętowych i programowych komputera. Można zatem powiedzieć, że system operacyjny tworzy środowisko programowe, które ma zapewnić realizację i współdziałanie programów użytkowych oraz wymianę informacji z otoczeniem w sposób poprawny, efektywny i niezawodny.

Rozróżnienie (klasyfikacja) systemów operacyjnych wynika przede wszystkim z przeznaczenia komputera, tj. typu problemu, jaki ma być rozwiązany przy jego użyciu. Na ogół rzeczywiste problemy charakteryzuje ich związek z otoczeniem, które je kreuje i których rozwiązanie przy użyciu komputera wpływa na stan otoczenia.

W tym kontekście systemy komputerowe można umownie podzielić na:

- systemy pracujące w trybie pośrednim (*off-line*), bez bezpośredniego sprzężenia z otoczeniem i braku uwarunkowań czasowych,
- systemy pracujące w trybie bezpośrednim (*on-line*), tj. sprzężone z otoczeniem bez uwarunkowań czasowych,
- systemy pracujące w trybie bezpośrednim czasu rzeczywistego (*real-time*), tj. sprzężone z otoczeniem z uwarunkowaniem czasowym.

Pojęcie systemu operacyjnego czasu rzeczywistego jest nieodłącznie związane z zastosowaniami komputerów, zwłaszcza do sterowania obiektami przemysłowymi.

Początki komputerowych systemów sterowania

Na koniec lat 60. i początek lat 70. przypada w Polsce okres pierwszych prób związanych z wprowadzaniem komputerów (minikomputerów) do sterowania wybranymi węzłami technologicznymi. Okres ten charakteryzował się przede wszystkim:

- brakiem ugruntowanych metod inżynierskiego podejścia do rozwiązania zadań związanych ze sterowaniem komputerowym.

- różnorodnością sprzętu komputerowego, często bez odpowiedniego oprogramowania podstawowego i narzędziowego.

Można tu wymienić minikomputery: typu RC 4000 firmy Reglen Zentralen, zakupione dla niektórych zakładów przemysłu azotowego, T2000/20 firmy Telemechanique, zakupione dla niektórych cementowni, H 316 firmy Honeywell, zakupiony dla przemysłu szklarskiego, czy też HP 2114 firmy Hewlett Packard, zakupiony dla jednej z hut.

Sytuacja na początku lat 70-tych była na tyle złożona, że problem efektywnego wykorzystania tych minikomputerów do sterowania ważnymi gospodarczo procesami technologicznymi postawiono aż na szczeblu rząd – PAN. W tym celu postanowiono powołać specjalny problem węzłowy 06.1.2. „Systemy automatyki kompleksowej", koordynację którego powierzono Zakładowi Systemów Automatyki Kompleksowej (ZSAK) PAN w Gliwicach.

Moment ten można przyjąć za początek zorganizowanych prac nad rozwiązywaniem problemów opartej na minikomputerach kompleksowej automatyzacji procesów technologicznych. W pracach tych wyróżniono następujące etapy:

- budowa modeli matematycznych procesów technologicznych i metody ich identyfikacji,
- synteza opartych na tych modelach matematycznych algorytmów sterowania,
- synteza struktur sterowania dla implementacji tych algorytmów,
- badania, a następnie praktyczna realizacja sterowania na obiektach rzeczywistych.

Wybranymi, przykładowymi węzłami technologicznymi były m.in.:

- piec obrotowy do wypalania klinkieru w cementowni,
- wanna szklarska w hucie szkła okiennego,
- reaktor syntezy amoniaku,
- walcarka taśmy w hucie,
- proces tlenowo-konwertorowy w hucie,
- wielki piec do produkcji surówki w hucie.

Związane z tym problemy informatyczne pojawiły się w momencie opracowania pierwszych algorytmów sterowania i podjęcia próby ich implementacji w minikomputerach.

Informatyka i mikroinformatyka czasu rzeczywistego

W końcu lat 60-tych pojęcie systemu operacyjnego było już dobrze ugruntowane w zastosowaniu do przetwarzania danych. Realizowano dwa podstawowe warianty przetwarzania:

- przetwarzanie wsadowe,
- przetwarzanie wielodostępne z podziałem czasu (ang. *time sharing*).

Podstawowym zadaniem takiego przetwarzania było realizowanie obliczeń, zgodnie z przygotowanym programem oraz w skończonym czasie. Czas ten starano się minimalizować przez odpowiednią organizację dostępu użytkowników do systemu oraz modyfikacje jego części sprzętowej (szybsze pamięci operacyjne) lub programowej (usprawnienie systemu operacyjnego).

Należy zaznaczyć, że tworzone łącznie ze sprzętem systemy operacyjne miały głównie na celu **optymalne wykorzystanie zasobów,** głównie procesora i pamięci. Awarie systemu nie były groźne, gdyż proces obliczeń można było powtarzać. Tworzone wówczas ośrodki obliczeniowe pełniły rolę usługową w odniesieniu do szeroko rozumianego przetwarzania danych. Sytuacja uległa diametralnej zmianie gdy zaczęto realizować ideę wyjęcia komputera z klimatyzowanego pomieszczenia i przeniesienia go w bezpośrednie otoczenie aparatury kontrolno-pomiarowej. W wyniku pojawienia się tego nowego kierunku zastosowań sprzętu komputerowego powstała tzw. **informatyka czasu rzeczywistego.** Charakteryzowała się ona próbami rozwiązania dwóch podstawowych problemów:

- konstrukcji minikomputerów predystynowanych do pracy w czasie rzeczywistym,
- syntezy systemów operacyjnych i oprogramowania narzędziowego umożliwiającego implementację algorytmów sterowania.

Konstrukcje minikomputerów musiały uwzględnić zmianę otoczenia, w którym miały one niezawodnie pracować. Stąd w ich strukturze pojawiły się takie elementy, jak **autonomiczny kanał przemysłowy,** umożliwiający kontakt pomiarowo-wykonawczy z obiektem, **rozbudowany układ przerwań** dla rejestracji zdarzeń w sterowanym obiekcie i jego otoczeniu, **zegar czasu rzeczywistego** dla synchronizacji pracy minikomputera z rytmem sterowanego obiektu.

Dla właściwej gospodarki tymi dodatkowymi zasobami minikomputera pojawił się problem syntezy specyficznego systemu operacyjnego, który przyjęto nazywać **systemem operacyjnym czasu rzeczywistego.** Można więc powiedzieć, że system operacyjny czasu rzeczywistego to program (zbiór programów) umożliwiający osiągnięcie celu sterowania obiektem przy ograniczeniach wynikających przede wszystkim z warunków czasu rzeczywistego, tj. zgodnie z dynamiką obiektu i otoczenia.

Problematykę syntezy i analizy takich systemów operacyjnych uznano za podstawę, poświęcając jej m.in. zorganizowane przez ZSAK PAN w latach 1973, 1974 i 1975 tzw. szkoły „Systemy operacyjne czasu rzeczywistego". Rozpoczęto także próby pisania systemów operacyjnych czasu rzeczywistego dla powstających w Polsce minikomputerów (K202, MKJ-25, ODRA 1310 SMA, MERA 400). W połowie lat 70. podsumowano wyniki uzyskane w problemie węzłowym 06.1.2. „Systemy automatyki kompleksowej". Okazało się, że trudno było wskazać obiekt sterowany automatycznie (w czasie rzeczywistym) przez minikomputer. Analiza przyczyn wskazywała z jednej strony na znaczne trudności w uzyskaniu odpowiednich modeli matematycznych obiektów i opartych na nich algorytmów sterowania, a z drugiej strony na potrzebę zmiany podejścia i konieczność tworzenia systemów operacyjnych czasu rzeczywistego wraz z odpowiednim oprogramowaniem narzędziowym ułatwiającym różnorodne aplikacje.

Ze względu na brak doświadczeń w zakresie tworzenia profesjonalnego oprogramowania podstawowego i narzędziowego przyjęto różne, lepiej lub gorzej udokumentowane, wzorce zachodnie, zarówno w zakresie rozwiązań sprzętowych, jak i oprogramowania systemowego. Stąd, wkrótce po zakończeniu problemu węzłowego 06.1.2 pojawiły się w różnych ośrodkach, nie zawsze związanych profesjonalnie z informatyką, konstrukcje wzorowane na rozwiązaniach takich firm jak Hewlett Packard czy Digital Equipment Corporation. Przy realizacji tych konstrukcji nacisk położono na zgodność rozwiązań sprzętowych z oprogramowaniem podstawowym pierwowzorów. Za przykład mogą posłużyć takie minikomputery, jak SMC-3, SMC-10 oraz MERA 60.

Sytuacja taka powtórzyła się na przełomie lat 70–80 wraz z pojawieniem się w kraju mikroprocesorów rodziny INTEL 8008/8080/ /8085, MCS-48 oraz Zilog Z80, a następnie w latach 80-tych wraz z pojawieniem się mikrokomputerów klasy IBM PC. Doświadczenia z lat poprzednich wykorzystano do konstruowania mikrokomputerów ogólnego przeznaczenia ComPAN-8 z dyskowym systemem operacyjnym CP/M, ComPAN-16 z systemem MS-DOS oraz mikrokomputerów przemysłowych MisterZ-80. Równolegle z konstruowaniem mikrokomputerów powstawały dedykowane układy mikroprocesorowe będące elementami urządzeń technicznych. O ile w przypadku mikrokomputerów istniały systemy operacyjne, o tyle projektowanie i wdrażanie układów dedykowanych wymagało opracowania środowiska wspomagającego uruchamianie zarówno sprzętu, jak i oprogramowania.

Zagadnienia te stały się przedmiotem **mikroinformatyki,** a w szczególności **mikroinformatyki czasu rzeczywistego.** W ramach tej dziedziny powstały oryginalne opracowania, tzw. emulatory, których znakomitymi przykładami są RTDS-8 i RTDS-16 [4] oraz programy uzdatniające (*debugery*) [4]. Jednocześnie prowadzono prace nad stworzeniem środowiska systemowego aplikacji czasu rzeczywistego wbudowanych w układy mikroprocesorowe lub wykonywanych w otoczeniu szeroko rozpowszechnionego systemu MS-DOS. Środowisko takie nazywano **monitorami czasu rzeczywistego** [12] lub **jądrami czasu rzeczywistego** [9].

Proste rozwiązania komputerowego sterowania

Charakteryzując otoczenie systemu komputerowego dla którego występują silne ograniczenia czasowe, należy stwierdzić, że otoczenie to, to przede wszystkim obiekty technologiczne, w których przebiegają złożone zjawiska fizyko-chemiczne z towarzyszącymi im strumieniami masy i energii. Zjawiska te cechuje właściwa im dynamika i każda próba oddziaływania na ich przebieg musi ją uwzględniać.

W odniesieniu do systemów komputerowych spowodowało to konieczność wprowadzenia mechanizmu reakcji na zaburzenia zewnętrzne, występujące w ich otoczeniu, które powinny modyfikować zachowanie się systemu komputerowego. Funkcję tę powierzono tzw. **układowi przerwań,** który stał się standardowym elementem struktury systemów komputerowych predestynowanych do zastoso-

wań w warunkach czasu rzeczywistego. Pełne uwzględnienie warunków czasu rzeczywistego wymagało jednak jego pomiaru w celu właściwego interpretowania stanów układu przerwań co w konsekwencji doprowadziło do umieszczenia w strukturze komputera zegara czasu rzeczywistego.

Różna była przy tym waga poszczególnych zadań występujących w otoczeniu systemu komputerowego. Aby to uwzględnić, wprowadzono pojęcie **priorytetu przerwania.** Przyjęto przy tym zasadę, że im mniejszy numer przerwania, np. 0, 1, 2, ... itd., tym wyższy jego priorytet. Wymagało to także przyjęcia określonej filozofii obsługi przez procesor tak rozumianego układu przerwań. Obsługa przerwania była rozumiana jako wywołanie i niezawodna realizacja tzw. **programu obsługi przerwania.** W procesie jego realizacji mogły jednak pojawiać się przerwania o wyższym priorytecie. Komplikowało to znacznie strukturę i funkcje systemu operacyjnego czasu rzeczywistego, który musiał zapewnić właściwą obsługę układu przerwań. Prowadziło to do **hierarchizacji** zdarzeń rejestrowanych przez układ przerwań i wprowadzenia tzw. regulaminów selekcji tych zdarzeń przed właściwą reakcją procesora na ich wystąpienie. Cel ten próbowano osiągnąć przez wprowadzenie tzw. regulaminów obsługi układu przerwań, wyróżniając np. **regulamin priorytetowy bezwzględny, regulamin priorytetowy względny, regulamin naturalny (FIFO).**

Sterowanie komputerowe oparte na przerwaniach

Mechanizmy te są charakterystyczne dla klasy systemów operacyjnych czasu rzeczywistego opartych na układzie przerwań. Dodatkową cechą tych systemów było także i to, że jakby malała rola czasu rzeczywistego jako miary dynamiki zdarzeń występujących w otoczeniu. Wynikało to z przyjętej zasady reakcji komputera wyłącznie na zdarzenia w jego otoczeniu (ang. *event driven*).

W efekcie prowadziło to do zbyt dużego uzależnienia czasu reakcji komputera od zdarzeń zewnętrznych, często zdominowanych różnorodnymi zakłóceniami generowanymi w kanałach przemysłowych. Zjawisko to nazwano „migotaniem" układu przerwań [13] i dotyczyło sytuacji, w której działanie systemu operacyjnego sprowadzało się do ciągłych prób obsłużenia do końca wybranego przerwania, co było jednak niemożliwe. Krótko mówiąc, system operacyjny reagował przez układ przerwań na znacznie więcej zdarzeń zewnętrznych niż tego wymagały dostępne algorytmy sterowania.

Próby ograniczania roli układu przerwań

Odejście od powszechnego wykorzystania dla kontaktu komputera z otoczeniem układu przerwań miało kilka kierunków:

- logiczne (semantyczne) maskowanie układu przerwań,
- filtracja i sprawdzanie wartości sygnałów wejściowych przy jednoczesnym doskonaleniu modeli matematycznych i algorytmów sterowania,
- wykorzystanie informacji o dynamice obiektu.

Maskowanie układu przerwań miało na celu blokowanie obsługi zdarzeń w trakcie realizacji obsługi zadań o wyższym priorytecie. Praktyka wykazała jednak ograniczoną przydatność tego sposobu obsługi układu przerwań.

Filtracja i sprawdzanie wartości sygnałów wejściowych miała na celu przede wszystkim eliminację zakłóceń powodowanych błędnym działaniem czujników, przetworników i torów pomiarowych lub wpływem na tor pomiarowy zakłóceń zewnętrznych, takich jak np. silne pola elektromagnetyczne, chwilowe spadki napięcia itp. Doskonalenie modeli matematycznych i opartych na nich algorytmów sterowania miało na celu próbę lepszego określenia warunków filtracji wartości sygnałów wejściowych, chociaż nie zawsze było to możliwe.

Wykorzystanie informacji o dynamice obiektu prowadziło czasem do poprawnego rozwiązania problemu reakcji systemu komputerowego na zdarzenia w otoczeniu.

Komputerowe sterowanie z cyklem periodycznym

U podstaw koncepcji systemu operacyjnego o cyklu periodycznym stanęła obserwacja sposobu zachowania się człowieka-operatora w procesie sterowania złożonymi obiektami technologicznymi. Podstawową cechą tego procesu była periodyczność dotycząca oceny przez niego sytuacji w obiekcie i podejmowania decyzji o sterowaniu. Problem dotyczył wyłącznie sposobu doboru cyklu zbierania i przetwarzania danych pomiarowych oraz określania wartości sterowania w przypadku sterowania komputerowego [15]. Po określeniu takiego cyklu dla konkretnego obiektu, realizacja sterowania komputerowego sprowadzała się do zliczania ciągu impulsów generowanych przez układ przerwań oraz aktywizacji odpowiednich programów zbierających dane pomiarowe z obiektu i określających (wykorzystując odpowiednie tablice decyzyjne) wartości sterowań.

W implementacji komputerowej prowadziło to do dwóch rozwiązań:

- segmentacji czasowej obsługi zdarzeń, której istota polegała na wyznaczeniu okresu periodycznej realizacji sekwencji programów oceny stanu obiektu oraz programów obliczających odpowiedzi sterujące,
- segmentacji pamięciowej obsługi zdarzeń, polegającej na uszeregowaniu w pamięci operacyjnej programów obsługi zdarzeń zgodnie z kolejnością ich wywoływania w jednym okresie cyklu akwizycji danych i określających odpowiedzi sterujące.

Komputerowe sterowanie oparte na rozbudowanym układzie przerwań, a także z cyklem periodycznym, mogło być stosowane z powodzeniem w przypadku niektórych obiektów technologicznych posiadających specyficzne własności zarówno w odniesieniu do ich dynamiki, jak też zmian w ich otoczeniu. Nie udało się jednak uogólnić tych doświadczeń i opracować uniwersalnej zasady komputerowego sterowania szeroką klasą złożonych obiektów technologicznych. W tej sytuacji postanowiono skoncentrować uwagę na dogłębnym poznaniu specyfiki uwarunkowań czasowych związanych ze sterowaniem w czasie rzeczywistym i na tej podstawie odpowiedzieć na pytanie, czym powinien być system operacyjny czasu rzeczywistego predystynowany do profesjonalnych zastosowań w systemach automatyki kompleksowej.

Wymagania czasu rzeczywistego

Jak już wspomniano, systemy komputerowe czasu rzeczywistego (ang. *real time computer systems*) wyróżniają się spośród innych typów przetwarzania uwzględnieniem czasu jako podstawowego parametru ich charakterystyki. Jedną z ogólnych, a zarazem najtrafniejszych definicji tego typu systemów podaje niemiecka norma przemysłowa [3]: „Jest to rodzaj systemu komputerowego, który umożliwia permanentne przetwarzanie nadchodzących z otoczenia danych tak, że wyniki przetwarzania są dostępne w określonych przedziałach czasu. W zależności od zastosowania, nadchodzenie danych następuje w przypadkowych lub określonych momentach czasu".

Ponieważ systemy te są zawsze związane z zastosowaniem (otoczeniem), nazywane są one także systemami wbudowanymi (ang. *embeded*) lub „zanurzonymi" w zastosowaniu (otoczeniu).

Stawiane systemom czasu rzeczywistego wymagania możemy sformułować w dwóch podstawowych kategoriach:

- współdziałanie z otoczeniem,
- zachowanie się samego systemu.

Te dwa typy wymagań odnoszą się do warunków czasowych jakie powinny systemy spełniać i są ze sobą związane. Współdziałanie z otoczeniem określa warunki czasowe: zdolność obserwacji (jednocześnie) zachodzących w otoczeniu zdarzeń oraz wyznaczenie właściwych momentów odpowiedzi systemu (oddziaływania na otoczenie). Wymaganie odnośnie zachowania się systemu dotyczy determinizmu i pewności jego działania w czasie wypracowywania odpowiedzi.

Krótko charakteryzując te systemy można powiedzieć, że dewizą systemów czasu rzeczywistego powinna być:

- gotowość do współdziałania,
- rzetelność,
- punktualność.

W środowisku czasu rzeczywistego takie podstawowe funkcje, jak zbieranie danych, przetwarzanie i wypracowanie odpowiedzi powinny zostać wykonane w odpowiednim, dla tego środowiska, czasie. Nie sama szybkość przetwarzania komputera jest zatem decydująca w systemie czasu rzeczywistego, lecz także zdolność reagowania w odpowiednim momencie czasu. Funkcjonalna poprawność systemu zależy nie tylko od poprawności wyników przetwarzania, ale również od momentu czasu, w którym wyniki te są dostępne. Właściwe momenty czasu są wyznaczane przez otoczenie systemu.

Przedstawione wyżej wymagania właściwie powinny odnosić się do każdego typu systemu komputerowego, lecz w przypadku systemów czasu rzeczywistego nabierają szczególnego znaczenia. Systemy tego typu pracują najczęściej w otoczeniu instalacji przemysłowych lub nadzorują urządzenia sterujące w środkach transportu. Spóźniona lub niepoprawna odpowiedź może spowodować taki sam, niebezpieczny, skutek. O ile w systemach komputerowych ogólnego przeznaczenia błędne lub spóźnione wyniki obliczeń są niepożądane, to w systemach czasu rzeczywistego reakcje błędne są szkodliwe lub wręcz niedopuszczalne.

Wymagania czasowe (ograniczenia czasowe) systemu czasu rzeczywistego możemy podzielić na dwie klasy:

- warunki ostre (ang. *hard real-time*),
- warunki słabe (ang. *soft real-time*).

O ile w przypadku słabego uwarunkowania czasowego można rozważać o kosztach spowodowanych naruszeniem dopuszczalnych ograniczeń, o tyle w przypadku uwarunkowania ostrego, niedopuszczalne jest w ogóle naruszanie tych ograniczeń. W otoczeniu systemu, w szczególnym przypadku zdarzenia pojawiają się jednocześnie, co prowadzi do stanu rywalizacji zgłoszeń o ich obsługę. W systemie może wystąpić pewnego rodzaju zaburzenie, które nie powinno naruszać dopuszczalnego ograniczenia.

Działanie systemu powinno być na tyle przewidywalne, aby programista mógł zaplanować czynności systemu operacyjnego prowadzące do poprawnej reakcji systemu, nie naruszając wymagań czasowych. Przewidywalność wystąpienia określonych zdarzeń i determinizm zachowania systemu umożliwiają opracowanie takiego postępowania, które w kłopotliwych sytuacjach doprowadzi do wypracowania przez system właściwej reakcji.

Krytyczne uwarunkowania czasowe występują np. w niższych warstwach systemu sterowania oraz w systemach nadzoru medycznego.

Powyższe uwagi prowadzą do pewnych ogólnych wniosków odnośnie cech oprogramowania oraz sprzętu systemów czasu rzeczywistego. Generalną zasadą tworzenia tego typu systemów powinna być prostota stosowanych rozwiązań [1] zgodnie ze schematem:

prostota ⇒ przewidywalność
zachowania ⇒ pewność działania

System operacyjny czasu rzeczywistego

Począwszy od pracy [1] klasycznym sposobem prezentacji systemu operacyjnego jest opis poszczególnych warstw tworzących hierarchiczny model jego struktury. Minimalny zestaw funkcji tworzy warstwę najniższą nazywaną **jądrem systemu** (ang. *kernel*). Kolejne wyższe warstwy powstają przez wykorzystanie funkcji warstw niższych, aż do warstwy obejmującej oprogramowanie użytkowe (aplikacyjne).

Charakterystykę jądra systemu czasu rzeczywistego, zachowując zasadę prostoty, przedstawiono w [8]. Tutaj cechy jądra wymieniamy z komentarzem z punktu widzenia omówionych wymagań stawianych systemom czasu rzeczywistego. Podstawową cechą jest współbieżność umożliwiająca równoczesne wykorzystywanie procesów obsługi asynchronicznie (przypadkowo) pojawiających się zdarzeń w otoczeniu. Cecha ta jest uzyskiwana w wyniku wykorzystania przez jądro funkcji przełączania procesora między procesami obsługi zdarzeń.

Kolejne cechy wynikają z warunku dotyczącego przewidywalności czasu reakcji systemu. Są to:

- **Wywłaszczalność.** Funkcja wywłaszczania procesu zapewnia dostęp do procesora procesowi o wyższym priorytecie niż ten, jaki posiada proces aktualnie realizowany. Mechanizm wywłaszczania pozwala oszacować czas reakcji systemu na zdarzenia o wysokim stopniu ważności.
- **Przełączanie kontekstu.** Przekazanie procesora procesowi o wyższym priorytecie (połączenie kontekstu) powinno być zrealizowane w możliwie najkrótszym i określonym czasie.
- **Opóźnienie w przyjęciu zgłoszenia (przerwania).** Podobnie jak poprzednio, wymaga się aby narzut czasowy związany z przyjęciem przerwania był minimalny. Czas blokowania układu przerwań nie powinien przekraczać ustalonego ograniczenia.
- **Synchronizacja i komunikacja.** Właściwa reakcja systemu wymaga najczęściej współdziałania wielu procesów realizujących zadanie użytkowe, które wypracowuje odpowiedź na zgłoszone zdarzenie. Jądro systemu powinno zatem dysponować mechanizmami synchronizacji i komunikacji między procesami (np. semafory, przesyłanie wiadomości).
- **Wydajność.** Podstawowym parametrem systemu komputerowego czasu rzeczywistego jest graniczna wydajność systemu określana w sytuacji skrajnie niekorzystnej konfiguracji zgłoszeń i obciążenia (najgorszy przypadek). Ocena tak rozumianej wydajności systemu metodami probabilistycznymi jest praktycznie nieprzydatna z punktu widzenia spełnienia wymagania dotyczącego przewidywalności zachowania się systemu.
- **Wymagania specjalne.** Cechą charakterystyczną systemu komputerowego czasu rzeczywistego jest traktowanie czasu jako informacji, podobnie do zmiennych stanu otoczenia. Wyposażenie systemu w układ sprzętowy rejestrujący czas wystąpienia zgłoszenia umożliwiałby ocenę rzeczywistego opóźnienia reakcji zadania aplikacyjnego na zdarzenie zewnętrzne, np. zmiana wartości zmiennej stanu. Parametr ten może mieć znaczenie w zastosowaniach sterowania otoczeniem z komputerem w torze sprzężenia zwrotnego. Istnieją algorytmy sterowania, które opierają się na stałej odległości w czasie między wartością sygnału wyjściowego obiektu sterowania a wartością sygnału sterującego. Ponadto znacznik czasu zarejestrowany w chwili reakcji na zgłoszenie można wykorzystać w algorytmie szeregowania i przydzielania procesora.

Dodatkową wymaganą funkcją systemu jest śladowanie stanu systemu, czyli rejestrowanie w celu późniejszej analizy [15], [16].

Przykład systemu operacyjnego czasu rzeczywistego

Dobrym przykładem systemu spełniającego wymagania czasu rzeczywistego jest system QNX firmy QUANTUM, opracowany dla mikrokomputerów z procesorami serii INTEL 8086/8088/80x86. QNX ma modularną strukturę. Moduły można podzielić na trzy typy: jądro, moduły zarządzające zasobami systemu (tzw. administratory) oraz programy użytkowe (aplikacje). Granica między administratorami a programami aplikacyjnymi jest najczęściej umowna, gdyż administratory mogą być opracowywane przez użytkownika systemu.

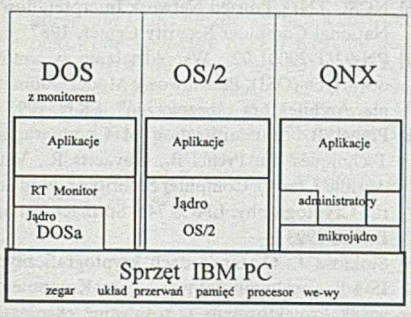
Jądro systemu jest małym modułem zapewniającym realizację na niskim poziomie podstawowych operacji dotyczących przechwytywania i kierowania przerwań do procesów obsługi, szeregowania procesów i przydzielania im procesora, obsługi interfejsu sieci oraz komunikacji między lokalnymi i rozproszonymi procesami. Implementacja jądra wymaga poniżej 10 kB pamięci. Opóźnienie związane z przekazaniem zgłoszenia przerwania wynosi od kilku do kilkunastu mikrosekund w zależności od typu procesora i częstotliwości zegara.

Administratory są podstawowymi modułami systemu. Można rozróżnić dwa typy administratorów: firmowe-podstawowe i opcjonalne oraz użytkowe (opracowane przez użytkownika systemu). System jest wyposażony w cztery podstawowe moduły firmowe: administratora procesów, urządzeń plików, oraz sieci. Głównym modułem jest administrator procesów, którego zadaniem jest: gospodarka pamięcią, tworzenie, usuwanie i zarządzanie procesorami, w tym administratorami, oraz zegarami (ang. *timers*). Jądro wraz z administratorem procesów tworzy system w najmniejszej konfiguracji. Pozostałe moduły są dołączane w zależności od potrzeb zastosowania. Administrator urządzeń obejmuje obsługę konsoli oraz sterownika portów równoległych i szeregowych mikrokomputera. Obsługa plików jest realizowana przez wielowątkowy proces administratora plików. W ten sposób umożliwiono współbieżne korzystanie z urządzeń plikowych wielu procesom żądającym obsługi.

Komunikacja pomiędzy procesami oraz *synchronizacja* ich współdziałania jest oparta głównie na przesyłaniu komunikatów (ang. *passing massages*) oraz kolejkach (ang. *queues*). Ponadto zaimplementowano porty (ang. *ports*) oraz wyjątki (ang. *exceptions*) jako mechanizmy komunikacji asynchronicznej. W systemie udostępniono trzy algorytmy szeregowania procesów, FIFO, Round-Robin oraz „sprawiedliwy", podobny do unixowego. Zmiana kontekstu procesu trwa od kilkunastu do kilkudziesięciu mikrosekund, w zależności od rodzaju procesora.

Usługi sieciowe są realizowane za pośrednictwem administratora sieci. Wymiana informacji przez sieć jest realizowana za pomocą

połączeń wirtualnych. Komunikacja między procesami w różnych węzłach przebiega dwuetapowo. Procesy komunikują się z końcami połączenia na zasadzie komunikacji międzyprocesowej. Administrator sieci realizuje przesłanie komunikatu z jednego do drugiego końca połączenia wirtualnego.



Struktura przykładowych systemów operacyjnych

Modularność systemu QNX umożliwia dostosowanie konfiguracji systemu do sprzętu użytego do konkretnego zastosowania. W minimalnym zestawie system wymaga ok. 150 kB pamięci i może rezydować w pamięci ROM. Dodatkową zaletą jest jego zgodność ze standardem POSIX, przynajmniej dla użytkowników UNIX-a.

W zastosowaniach mniej krytycznych czasowo alternatywą dla QNX-a może być system OS/2 firmy IBM. OS/2 jest wielowątkowym systemem o strukturze wielowarstwowej, w skład którego wchodzi jądro, programy obsługi urządzeń, łączone dynamicznie biblioteki (ang. *dynamic link libraries*) oraz programy użytkowe [2]. W porównaniu do systemu QNX ma on duże jądro oraz znacznie większe czasy blokowania przerwań i przełączania kontekstu. Bogaty zestaw mechanizmów synchronizacji oraz komunikacji między procesorami pozwala implementować wielozadaniowe aplikacje lokalne i rozproszone.

Zarówno system QNX, jak i OS/2 mogą służyć za platformę aplikacji z uwarunkowaniami czasu rzeczywistego.

LITERATURA

[1] Brinch Hansen: The Nucleus of Multiprogramming System. Communication of ACM, 13, 4, 1970
[2] Deitel H.M., Kogan M.S.: The Design of OS/2. Addision-Wesley Publishing Company, 1992
[3] DIN 44 300: Informationsverarbeitung, no 9.2.11. Beuth-Verlag, Berlin, 1985
[4] Gorawski M., Gabryś B., Winiarczyk R.: On the Industrial Phase of the Development of a Process Control Microcomputer System. Microprocessing and Microprogramming, v. 15, no. 2, 1985
[5] Grega W., Kołek K.: Ekonomiczne realizacje układów sterowania czasu rzeczywistego na IBM PC. Informatyka nr 4, 1994
[6] Halang W.A.: Contemporary Computers Considered Inappropriate for Real-Time Control, Algorithms and Architectures for Real-Time Control. IFAC Workshop, 1992
[7] Mrózek A.: Modele wnioskowania operatorów i ich wykorzystanie w komputeryzacji obiektów technologicznych. Praca habilitacyjna, Gliwice, 1987
[8] Nałęcki K.: Dlaczego UNIX nie jest systemem czasu rzeczywistego?, INFORMATYKA nr 6, 1991
[9] Płonka L.: The RTC Real-Time Multitasking Executive. Raport IITiS PAN, Gliwice, 1993
[10] Pyle I., Hruschka P., i inni: Real-Time Systems Investigating Industrial Practice. John Wiley and Sons, 1993
[11] Silberschatz A., Peterson J.L., Galvin P.B.: Podstawy systemów operacyjnych. PWNT, 1993
[12] Trojniar W.: Metody implementacji systemów czasu rzeczywistego. INFORMATYKA nr 6, 1991
[13] Yourdon E.: Projektowanie systemów o działaniu bezpośrednim. WNT, Warszawa, 1976
[14] Zonenberg D.: Emulator układowy mikroprocesorów jako narzędzie uruchomieniowe w ujęciu programistycznym. INFORMATYKA nr 2, 1992
[15] Zonenberg D.: Emulator układowy – obserwacja danych do analizy zachowań systemów czasu rzeczywistego. Zeszyty Naukowe Politechniki Śląskiej, seria Informatyka, z. 22, 1993
[16] Zonenberg D.: Emulator układowy – obserwacja uzależnień czasowych oprogramowania systemu mikroprocesorowego. Zeszyty Naukowe Politechniki Śląskiej seria Informatyka, z. 22, 1993

Modele zbiorów rozmytych... Część 2

dokończenie ze s. 22

ich podstawie konstruować reguły wnioskowania, np. przy wykorzystaniu przedstawionego wyżej algorytmu indukcyjnego.

Tabela 5. Wartości lingwistyczne, wartości wybranych kryteriów i wskaźników oceny działalności przedsiębiorstw

Wybrane wskaźniki oceny działalności przedsiębiorstwa	Wartości lingwistyczne wskaźników oceny		
	mała M	średnia S	duża D
Liczba ofert na zamówienia w przetargu	< 10	10–25	> 25
Liczba otrzymanych zamówień w przetargu	< 5	5–10	> 10
Wysokość narzutu	< 9,4	9,4–10,0	> 10,0
Wartość realizowanych zamówień w jednostkach pieniężnych	< 2000	2000–4000	> 4000
Wartość zysku przedsiębiorstwa w jednostkach pieniężnych	< 60	60–120	> 120
Wartość współczynnika sukcesu w przetargu	< 0,35	0,50	> 0,50

Przy modelowaniu działalności w określonych warunkach funkcjonowania przedsiębiorstwa, tj. parametrach gry decyzyjnej określanych przez jej Centrum, takich jak [5]: polityka spłaty kredytów i oprocentowanie, podatki od dochodów, wielkość narzutów od kosztów wykonania zamówień itp., główny wpływ na wynik finansowy mają koszty własne realizacji zamówień, wartość wykonania zamówień przez podwykonawców, a także koszty związane z przygotowaniem techniczno-ekonomicznych ofert.

Na podstawie przedstawionych w tab. 4 (przy określonych parametrach gry) wyników można stwierdzić, że:

- podstawowy wpływ na otrzymanie zamówień mają koszty ich realizacji, ponieważ dana oferta jest konkurencyjna nawet przy stosunkowo wysokich narzutach,
- aby osiągnąć wysoki zysk wskazane jest uzyskanie dużej liczby zamówień, a także przekazanie ich podwykonawcom, w przypadku ograniczonej własnej mocy produkcyjnej,
- podobne wyniki finansowe można uzyskać zarówno przy dużej wartości przyjętych zamówień i niskim narzucie, jak i przy wysokim narzucie i małej wartości zamówień itp.

Wszystkie wyżej przedstawione modele wspomagania decyzji wraz z modelem gry zostały zaprogramowane w języku Fortran i C oraz uruchomione na komputerze IBM/PC/AT.

Przedstawiona gra decyzyjna jest wykorzystywana w celu zapoznania jej uczestników z podstawowym mechanizmem gospodarki rynkowej określonym przez relację: podaż, popyt, cena. Służy ona także do modelowania skutków podejmowanych decyzji w warunkach niepewności.

LITERATURA

[1] Gawriłowa T.A., Czerwińskaja K.R.: Izwljeczjenie i strukturirowanije znanij dlja ekspertnych sistjem. Radio i Swjaz, Moskwa 1990
[2] Kacprzyk J.: Zbiory rozmyte w analizie systemowej. PWN, Warszawa 1986
[3] Pozyskiwanie wiedzy (przekł. z japońskiego). Mir, Moskwa 1990
[4] Witkowski T.: Pozyskiwanie wiedzy i konstrukcja reguł wnioskowania dla oceny prawdopodobieństwa sukcesu w przetargu. Materiały Krajowej Konferencji „Analiza decyzyjna. Systemy eksperckie. Zastosowania systemów komputerowych". Warszawa, 25–27 maja 1994 r. Instytut Badań Systemowych PAN, Warszawa 1994
[5] Witkowski T., Timaszowa L.A.: Modele oceny podejmowania decyzji finansowo-ekonomicznych przedsiębiorstw w warunkach ryzyka, w: Finansowo-ekonomiczne systemy wspomagania decyzji w kompleksowym zarządzaniu produkcją. Instytut Cybernetyki AN Ukrainy, Kijów 1993.

Architektura otwartego systemu do syntezy obrazów realistycznych

dokończenie ze s. 7

[75] Whitted T., Weimer D.: A Software Testbed for the Development of 3D Raster Graphics Systems. ACM Trans. Graph., 1 (1) 1982, pp. 43–58

[76] Wojdała A.: Praktyczna metoda śledzenia promieni dla systemów wieloprocesorowych. Machine Graphics & Vision, Vol. 1, No. 1/2 (Proceedings of the 2nd Conference GKPO 1992, Nałęczów), pp. 226–235

[77] Wojdała A.: Metoda śledzenia promieni: obecny stan wiedzy. Machine Graphics & Vision, Vol. 1, No. 1/2 (Proceedings of the 2nd Conference GKPO 1992, Nałęczów) pp. 108–155

[78] Wojdała A., Myszkowski K., Rej P.: Wspomaganie sprzętowe syntezy realistycznych obrazów. Machine Grapics & Vision, Vol. 1, No. 1/2 (Proceedings of the 2nd Conference GKPO 1992, Nałęczów), pp. 236–246

[79] Woo A.: Accelerators for shadow determination in ray tracing, M.Sc. thesis. Dept. of Comp. Science, Univ. of Toronto.

O standaryzacji w zakresie technik kryptograficznych

dokończenie ze s. 15

[23] ISO/IEC 10736 „Information Technology – Open System Interconnection – Transport Leyer Security Protocol. ISO/IEC

[24] ISO/IEC 11577 „Information Technology – Telecommunications and Information Exchange Betwen Systems – Network Layer Security Protocol. ISO/IEC

[25] ITSEC „Information Technology Security Evaluation Criteria. Provisional Harmonised Criteria. Ver. 1.2, 1991

[26] NCSC TNI „Trusted Network Interpretation". National Computer Security Center, 1987

[27] PN-92/T-20001/02 „Współdziałanie systemów otwartych (OSI). Podstawowy Model Odniesienia. Architektura zabezpieczeń". PKN, 1993

[28] Prenel B.: Standardization of Cryptographic Techniques. In: Prenel B., Govaerts R., Vandewalle J. (eds.): Computer Security and Industrial Cryptography. LNCS 741, Springer-Verlag, Berlin, 1993

[29] Stokłosa J.: O standardach kryptograficznych ISO dla sieci komputerowych. Sieci Komputerowe'89. Projektowanie, zastosowanie, eksploatacja. Wydawnictwo Politechniki Wrocławskiej, Wrocław, 1989, s. 237–240.

Wojdała A.: Architektura otwartego systemu do syntezy obrazów realistycznych
INFORMATYKA 1994, nr 9, s. 3
Charakterystyka architektury otwartego systemu syntezy realistycznych obrazów, przeznaczonego do prac badawczych i dydaktyki.

Wojdała A.: An open image synthesis system architecture
INFORMATYKA 1994, No. 9, p. 3
Characteristics of an open system architecture for realistic image synthesis, aimed at research and didactics.

Wojdała A.: Architektur eines offenen Systems zur Synthese der realistischen Bilder
INFORMATYKA 1994, Nr. 9, S. 3
Eine Charakteristik von Architektur eines offenen Systems zur Synthese der realistischen Bilder, das für Forschung und Didaktik bestimmt ist.

Dziuba D., Elsner T., Lasek M., Scheer A.W.: ARIS – Architektura Zintegrowanych Systemów Informacyjnych
INFORMATYKA 1994, nr 9, s. 8
Metodyka i zestaw programów do modelowania i realizacji zintegrowanych systemów informatycznych, opracowana na Uniwersytecie Saarbrücken.

Dziuba D., Elsner T., Lasek M., Scheer A.W.: ARIS – Architecture of Integrated Information Systems
INFORMATYKA 1994, No. 9, p. 8
Methodics and program set for modeling and realization of integrated informatic systems, elaborated at Saarbrücken University.

Dziuba D., Elsner T., Lasek M., Scheer A.W.: ARIS – Architektur Integrierter Informationssysteme
INFORMATYKA 1994, Nr. 9, S. 8
Methodik und Programmensatz zur Modellierung und Realisierung der integrierten Informatiksystemen, die in der Universität Saarbrücken erarbeitet wurden.

Stokłosa J.: O standaryzacji w zakresie technik kryptograficznych
INFORMATYKA 1994, nr 9, s. 13
Przegląd narodowych i międzynarodowych organizacji standaryzacyjnych zajmujących się problemami kryptografii oraz aktualny stan opracowania dokumentów standaryzacyjnych w dziedzinie technik kryptograficznych.

Stokłosa J.: On standardization in cryptographic technics
INFORMATYKA 1994, No. 9, p. 13
Survey of national and international standard organizations engaged in cryptography problems and actual situation of standardization work in the field of cryptographic technic.

Stokłosa J.: Über Standardisierung im Bereich der kryptographischen Techniken
INFORMATYKA 1994, Nr. 9, S. 13
Ein Übersicht von nationalen und internationalen Standardenorganisationen, die sich mit Kryptographieproblemen beschäftigen, sowie heutiger Stand der Standardendokumentebearbeitung im Bereich der kryptographischen Techniken.

Stabrowski M.: Podprogramy w języku symulacji systemów dynamicznych AMIL
INFORMATYKA 1994, nr 9, s. 16
Implementacja mechanizmów definiowania, kompilacji, konsolidacji i wykonywania programów w języku symulacji systemów dynamicznych AMIL.

Stabrowski M.: Subprograms in the AMIL language for dynamic systems simulation
INFORMATYKA 1994, No. 9, p. 16
Implementation of programs definition, compilation, consolidation and execution in the AMIL language for dynamic systems simulation.

Stabrowski M.: Unterprogramme in AMIL – einer Sprache zur Simulation der dynamischen Systeme
INFORMATYKA 1994, Nr. 9, S. 16
Implementation von Mechanismen der Programmedefinierung, -kompilierung, -konsolidierung und -ausführung in der zur Simulation der dynamischen Systeme angepassten AMIL-Sprache.

Witkowski T.: Metody zbiorów rozmytych w analizie decyzyjnej przedsiębiorstwa. Część 2
INFORMATYKA 1994, nr 9, s. 20
Druga część przeglądu metod i modeli zbiorów rozmytych, stosowanych w procesach decyzyjnych przedsiębiorstwa.

Witkowski T.: Fuzzy sets methods in enterprise's decision analysis. Part 2
INFORMATYKA 1994, No. 9, p. 20
Second part of a preview of fuzzy sets methods and models applied in enterprise's decision process.

Witkowski T.: Fuzzy Sets-Methoden in Unternehmensbeschlussanalyse. Teil 2
INFORMATYKA 1994, Nr. 9, S. 20
Zweiter Teil einer Übersicht von Fuzzy Sets-Methoden und -Modellen, die im Unternehmensbeschlussprozess angewendet werden.

Mrózek A., Winiarczyk R.: Systemy operacyjne czasu rzeczywistego
INFORMATYKA 1994, nr 9, s. 23
Ewolucja pojęcia oraz specyfika wymagań systemów operacyjnych czasu rzeczywistego, a także przykłady takich systemów dla komputerów osobistych.

Mrózek A., Winiarczyk R.: Real-time operating systems
INFORMATYKA 1994, No. 9, p. 23
Evolution and specific requirements of real-time operating systems, as well as examples of such systems for personal computers.

Mrózek A., Winiarczyk R.: Echtzeitbetriebssysteme
INFORMATYKA 1994, Nr. 9, S. 23
Evolution des Begriffes und spezifische Forderungen an Echtzeitbetriebssysteme, sowie Beispiele von solchen Systemen für Personal computer.

*Oracle wprowadza na rynek nowy pakiet Oracle Server Manager, ułatwiający administratorom zarządzanie bazą danych. Server Manager to w praktyce dwa oddzielne narzędzia – jedno wyposażone w graficzny interfejs użytkownika (GUI Server Manager), drugie zaś (przeznaczone dla terminali tekstowych) działające z poziomu linii poleceń (Linemode Server Manager). Oba narzędzia są kompatybilne z SQL*DBA. GUI Server Manager ma 16 różnych funkcji monitorujących stan systemu. Za pomocą tego narzędzia m.in. można szybko konstruować nawet bardzo skomplikowane komendy SQL i wysyłać dowolne polecenia SQL lub SQL*DBA do dowolnej bazy danych. Server Manager jest wprowadzany na rynek wraz z systemem Oracle7 Production Realase 7.1 natomiast GUI Server Manager będzie dostępny początkowo jedynie na platformie Motif2.1, a następnie na Windows 3.1. Wersja Linemode jest integralną częścią nowego zestawu Oracle7. Obydwa produkty działają z każdą dostępną wersją Oracle7.*

* * *

Wartość sprzedaży Compaqa w 1993 r. wyniosła 7,2 mld USD, co oznacza wzrost w stosunku do 1992 r. o 75%. Wyniki sprzedaży Compaqa za I kwartał 1994 r. wyniosły 2,3 mld USD, co oznacza wzrost o 41% w stosunku do analogicznego okresu ub.r. (za cały 1993 r. – 7,2 mld USD). Zysk netto za I kwartał br. wyniósł 213 mln USD i wzrósł w stosunku do I kwartału 1993 r. o 108%. Jest to wyraźnie związane z ofensywą produkcyjną firmy: w 1993 r. Compaq wprowadził na rynek 36 nowych modeli, w pierwszych czterech miesiącach br.r. – już 26 modeli, sprzedał zaś w ub.r. 3,1 mln komputerów, z tego w Polsce 778 tys. sztuk. Obrót na polskim rynku wzrósł w porównaniu z I kwartałem 1993 r. o 144%.

* * *

Zysk netto korporacji IBM w I kwartale br. wyniósł 392 mln USD. W analogicznym okresie ub.r. firma zanotowała stratę w wysokości 399 mln USD. Dochody brutto w I kwartale br. wyniosły 13,4 mld USD. Dochód brutto w I kwartale w USA wzrósł w stosunku do analogicznego okresu ub.r. o 6%, w Europie – o 3%, w Azji – o 13%, w Ameryce Łacińskiej – o 14%. IBM zakończył I kwartał br. z gotówką i papierami wartościowymi o przybliżonej wartości 9 mld USD, przed rokiem w kasie korporacji znajdowało się tylko 5,8 mld USD.

* * *

Obroty największego brytyjskiego koncernu informatycznego ICL wzrosły w 1993 r. w stosunku do 1992 r. o 6% osiągając 2,6 mld funtów. Największy wzrost dochodów ICL odnotował w działalności serwisowej oraz rozwiązaniach informatycznych dla handlu detalicznego. Działająca od 30 lat ICL Poland osiągnęła w 1993 r. obroty w wysokości 530 mld zł. W cenach stałych oznacza to wzrost sprzedaży o 25% w stosunku do roku 1992. Klienci ICL Poland w 1993 r. to m.in. kilkanaście banków, URM, dwa ministerstwa, Fiat Auto Poland, zakłady przemysłowe, sieć marketów GLOBI. W ub.r. ICL przejął 51% udziałów w polskiej firmie SOFTBANK produkującej oprogramowanie dla bankowości. ICL Poland jest pierwszą firmą informatyczną w Polsce, która uzyskała certyfikat ISO 9001.

Prosto z firm • Prosto z firm • Prosto z firm • Prosto z firm

PIERWSZY POLSKI INTELIGENTNY
SYSTEM INFORMATYCZNY WSPOMAGAJĄCY ZARZĄDZANIE
PRODUKCJĄ - (TPPiRP), FINANSAMI - KOSZTAMI, SPRZEDAŻĄ, KADRAMI
WRAZ Z SYSTEMEM INFORMACJI KIEROWNICTWA
WSPOMAGANY SYSTEMEM SZTUCZNEJ INTELIGENCJI PC-SHELL
ORAZ EKSPERCKĄ BAZĄ WIEDZY EKONOMICZNO-FINANSOWEJ - ISAF
(DLA DUŻYCH PRZEDSIĘBIORSTW PRODUKCYJNYCH)

Perfect-Ekspert AITECH

(DZIAŁAJĄCY W OPARCIU O ŚRODOWISKO PRACY - WYŁĄCZNIE SYSTEM OPERACYJNY DOS, NetWare v3.11, v4.01 FIRMY NOVELL)

OFERUJE

PRZEDSIĘBIORSTWO PRODUKCYJNO-USŁUGOWE INNOWACJI TECHNICZNYCH

JUNISOFTEX Sp. z o.o.

ROK ZAŁOŻENIA 1987

WARTOŚĆ SPRZEDAŻY wg F-02 1992 (14,6 mld), wg F-01 na 1993.10.31 (15,4 mld)

TEL/FAX 315-131, 315-700, 321-525, 388-052, 388-054 TLX 036233

BIURO TECHNICZNO-HANDLOWE

44-100 GLIWICE ul. Kościuszki 1

WSZYSTKICH ZAINTERESOWANYCH ZAPRASZAMY NA MERYTORYCZNE ROZMOWY DO NASZEGO BIURA TECHNICZNO-HANDLOWEGO

OFERUJEMY RÓWNIEŻ INTELIGENTNE NARZĘDZIE DO PRODUKCJI EKSPERCKICH BAZ WIEDZY

PC-SHELL

ARTIFICIAL INTELLIGENCE

DLA JEDNOSTEK BUDŻETOWYCH (NAUKA, OŚWIATA, URZĘDY MIAST I GMIN, SŁUŻBA ZDROWIA) BEZ PRAWA DO KOMERCYJNEGO WYKORZYSTYWANIA W CENIE 5.000.000,- (BEZ VAT) - INFORMACJE I ZAMÓWIENIA PROSIMY SKŁADAĆ BEZPOŚREDNIO U PRODUCENTA ARTIFICIAL INTELLIGENCE LABORATORY - AITECH, KATOWICE 14 SKR. POCZT. 2804.

DLA POZOSTAŁYCH JEDNOSTEK GOSPODARCZYCH BEZ PRAWA KOMERCYJNEGO WYKORZYSTYWANIA TJ. WYŁĄCZNIE NA WŁASNE POTRZEBY DO KOŃCA I KW. 1994 ROKU W CENIE PROMOCYJNEJ 10.000.000,- (BEZ VAT), DLA KUPUJĄCYCH DO CELÓW KOMERCYJNYCH DO KOŃCA I KW. 1994 25.000.000,- (BEZ VAT), LICENCJA OGRANICZONA NA NIEOKREŚLONĄ ILOŚĆ ODBIORCÓW KOŃCOWYCH.

POSZUKUJEMY DO WSPÓŁPRACY LUB NAWIĄŻEMY WSPÓŁPRACĘ Z EKSPERTAMI Z DOWOLNYCH DZIEDZIN NAUKI I TECHNIKI.

0/3/93

Stacje Intergraph z Windows NT

Intergraph, znany światowy producent oprogramowania CAD, zdecydował się na przeniesienie większości swoich aplikacji CAD-owskich do środowiska Windows NT. Wiąże się to z podjęciem produkcji nowej generacji wysokowydajnych stacji roboczych tej firmy, opartych na procesorach Pentium. Środowisko Pentium/Windows już dziś oferuje szeroką gamę aplikacji od Office Automation po CAD. Jednolita architektura procesora zapewnia zgodność dzisiejszych programów z ich przyszłymi wersjami, gwarantując bezpieczeństwo inwestycji w sprzęt komputerowy i jego programowanie. Jako ciekawostkę można dodać, że dwuprocesorowa stacja robocza integraphu podczas wystawy Windows World posłużyła Billowi Gates, właścicielowi Microsofta, do prezentacji grafiki OpenGL, zainplantowanej do nowego systemu operacyjnego Daytona.

Biblioteka Watykańska

Biblioteka Watykańska, Pontyfikalny Uniwersytet Katolicki w Rio de Janeiro w Brazylii (PUC-Rio) oraz Korporacja IBM utworzyły projekt badań nad stworzeniem elektronicznego dostępu do wybranych zasobów Biblioteki Watykańskiej dla studentów i wykładowców z całego świata. Projekt ma wykorzystać najnowsze osiągnięcia z dziedziny cyfrowego magazynowania i kompresji danych, przetwarzania obrazu oraz telekomunikacji. Zespół IBM rozpocznie prace nad projektem komputeryzacji od wczytywania oryginalnych materiałów Biblioteki za pomocą cyfrowego skanera. Skaner, sprzężony ze stacją roboczą, będzie wykorzystywał opatentowane urządzenie TDI CCD (Time-Delay and Integration Charged-Coupled Device). Przestrzenna rozdzielczość skanowania może wynosić do 3072 × 4000 punktów, zaś rozdzielczość koloru – do 36 bitów na punkt.

Bezpieczeństwo baz danych Oracla

*Oracle opracował nowy system – Oracle Secure Network Services – oferujący kilka technik zabezpieczania, zapewniających konfidencjalność i integralność danych przesyłanych siecią. Produkt ten, uzupełniający oprogramowanie sieciowe Oracle SQL*Net, stworzony przy współpracy z firmą RSA Data Security, która pełni wiodącą rolę na rynku technologii zabezpieczania informacji, zawiera licencyjne techniki kodowania RC4, MD5 i Diffie -Hellman Oracle. System może być instalowany praktycznie na każdej platformie zamiast drogich kart kodujących. Alternatywą dla Oracle Secure Network Services jest adapter Oracle DCE, który integruje możliwości OSF DCE w zakresie zabezpieczania i łączności z opcjami klient-serwer produktów Oracle. Z pomocą DCE można zbudować aplikacje działające w środowisku rozproszonym, korzystające z wspólnego dla systemu operacyjnego i baz danych zestawu zabezpieczeń i transportu sieciowego. Oprócz integracji z systemami bezpieczeństwa DCE, Oracle DCE Adapter pozwala także na korzystanie bezpośrednio z grup zdefiniowanych w DCE Security Registry oraz na zadeklarowanie odpowiadających im ról w bazie danych. Umożliwia to zdefiniowanie klasy użytkowników posiadających takie same prawa w systemie operacyjnym jak i w serwerze. Obydwa produkty – w wersji beta – przechodzą obecnie testy w USA. Wersje produkcyjne mają być gotowe pod koniec br.*

Nowe inwestycje Compaqa

Compaq zamierza zainwestować kolejne 15 mln USD w dalszy rozwój swojego zakładu produkcyjnego w Erskine koło Glasgow. Pieniądze te przeznaczone są na dodanie trzech linii montażowych nowych płytek obwodów drukowanych (PCB – Printed Circiut Board) oraz utworzenie 300 nowych miejsc pracy. W styczniu br. firma zainwestowała w budowę dwóch linii PCB 10,5 mln USD. Pięć nowych linii PCB w Erskine będzie wykorzystywać najnowszą technologię montażu powierzchniowego (SMT – Surface-Mount Technology). Jak dotąd Compaq European Manufacturing zainwestował w Europie 130 mln USD. Zakład w Erskine o powierzchni 52 tys. m² zatrudnia obecnie ponad 800 osób i jest głównym dostawcą kompletnej linii produktów PC firmy Compaq na Europę, Afrykę i Bliski Wschód. Zakład ten otrzymał Nagrodę Królowej za osiągnięcia w dziedzinie eksportu za 1992 r. i był pierwszą fabryką Compaqa, której przyznano certyfikat jakości ISO 90002/BS5750. Compaq buduje fabrykę w Chinach, inwestuje dalsze 20 mln USD w fabrykę w Huston (USA), rozpoczyna produkcję desktopów w Brazylii i zapowiada, że do 1996 r. podwoi produkcję zakładu w Singapurze. Ta ekspansja Compaqa to efekt sukcesów finansowych firmy. W 1993 r. obroty firmy wyniosły 7,2 mld USD, z czego w Europie 2,7 mld USD. W połowie sierpnia br. Compaq ponownie obniżył ceny na swoje wyroby (o ok. 22%, a w Polsce nawet o ok. 30%). Jak oświadczyli szefowie firmy, podjęcie tej decyzji było możliwe dzięki wysokim obrotom. Czyżby nowe posunięcie w podjętej przez Compaqa przed dwoma laty wojnie cenowej?

Microsoft Project 4.0

Nowa wersja Microsoft Project 4.0 – program Windows do zarządzania projektami – dostępna jest także dla Macintosha. Pozwala on na planowanie, zarządzanie projektami oraz komunikację między grupami użytkowników. Podobnie jak produkty z zestawu Microsoft Office, nowy Project oferuje technikę IntelliSense. Dzięki temu wyczuwa on intencje użytkownika i podpowiada właściwe rozwiązanie. Microsoft Project umożliwia wykorzystywanie poczty elektronicznej w celu ułatwienia przekazywania danych dotyczących projektu. Jest też systemem programowalnym; jego funkcjonalność można rozszerzyć za pomocą wbudowanego języka programowania Visual Basic, Application Edition, będącego wspólnym językiem programowania aplikacji z zestawu Office. OLE Automation pozwala łączyć projekty z innymi aplikacjami Windows. Program będzie dostępny w ośmiu wersjach językowych, nie przewidziano na razie polskiej wersji. Cena katalogowa wynosi 695 USD, Upgrade od wersji 3.0 – 149 USD.

Opracowała:
KRYSTYNA KARWICKA-RYCHLEWICZ

SunFastEthernet 10-100 Mb/s

Pierwszą kartę sieciową, którą można instalować w stacjach roboczych systemu Unix – FastEthernet standardu 100Base-T – wprowadza na rynek Sun Microsystems Computer Co. Sprawdza ona z jaką szybkością pracuje podłączona do niej sieć i przystosowuje się do niej automatycznie przełączając swoje obwody na daną szybkość (zarówno 10 jak i 100 Mb/s). Wbudowany w kartę tranceiver 100Base-T pracuje z okablowaniem UTP Category 5, zaś interfejs MII pozwala współpracować karcie (po podłączeniu do zewnętrznego tranceivera) z innymi typami okablowania. Karta pracuje zgodnie ze standardem CSMA /CD (wykrywanie kolizji w sieci), co pozwala na stosowanie jej w już istniejących sieciach bez wymiany okablowania. SunFastEthernet zgodna jest też ze specyfikacjami zaakceptowanymi przez organizację Fast Ethernest Alliance, co oznacza, że jest kompatybilna z produktami sieciowymi firm będących członkami tej organizacji.

Oracle Transparent Gateway

*Dostępna jest już wersja produkcyjna zestawu Oracle Transparent Gateway for IBM DRDA. Pracuje on na platformie IBM RISC System/6000; a połączenia są realizowane przez SQL*Net 2. System działa bez konieczności instalacji produktów Oracle na jakiejkolwiek platformie docelowej. W momencie, gdy rozkaz języka SQL zostaje dostarczony do Oracle Transparent Gateway, jest automatycznie przetwarzany na transakcję DRDA i przesyłany przez SNA do bazy docelowej. Produkt Oracle jest zgodny ze zdefiniowanym przez IBM protokołem Remote Unitof Work, znanym także pod nazwą DRDA poziom 1. Oracle Transparent Gateway pozwala na wykonywanie rozproszonych zapytań sięgających do danych znajdujących się jednocześnie na kilku serwerach DRDA lub Oracle. Pozwala zarówno na formułowanie zapytań, jak i dokonywanie modyfikacji danych przechowywanych na wszystkich typach serwerów IBM DRDA: DB2/MVS, SQL/DS i AS/400 DBMS.*